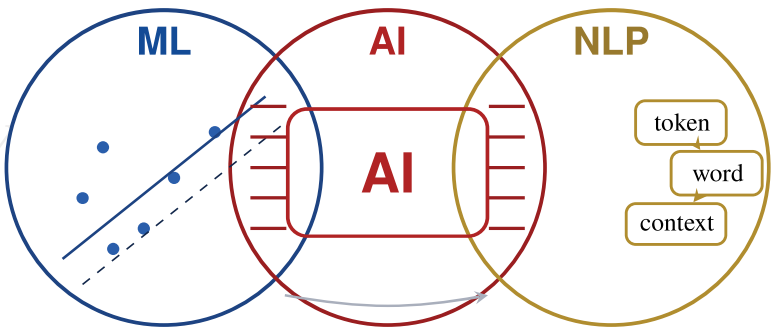
ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN

PGS.TS Huỳnh Công Pháp

ĐẠI SỐ TUYẾN TÍNH VÀ ỨNG DỤNG

TRONG TRÍ TUỆ NHÂN TẠO VÀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

(Sách chuyên khảo)



NHÀ XUẤT BẢN
KHOA HỌC - CÔNG NGHỆ - TRUYỀN THÔNG

Số xác nhận đăng ký xuất bản: 567-2026/CXBIPH/01-28/KHCNTT.
Số quyết định xuất bản: Số: 12/QĐ-NXBKHCNTT ngày 10 tháng 03 năm 2026.
Nộp lưu chiểu Quý I/2026. **ISBN: 978-604-45-2627-0**

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN

PGS.TS Huỳnh Công Pháp

ĐẠI SỐ TUYẾN TÍNH VÀ ỨNG DỤNG

TRONG TRÍ TUỆ NHÂN TẠO VÀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

(Sách chuyên khảo)

NHÀ XUẤT BẢN KHOA HỌC - CÔNG NGHỆ - TRUYỀN THÔNG

LỜI NÓI ĐẦU

Trí tuệ nhân tạo (AI) đang trở thành công cụ hỗ trợ đắc lực cho tất cả chúng ta trong đời sống hiện đại: từ gợi ý nội dung, nhận diện hình ảnh, xe tự hành, cho đến hệ thống hỗ trợ quyết định trong y tế, tài chính và giáo dục. Xử lý ngôn ngữ tự nhiên (NLP) giữ vai trò đặc biệt trong số các nhánh ứng dụng trí tuệ nhân tạo. Ngôn ngữ tự nhiên là phương tiện cơ bản và thiết yếu để con người tư duy, giao tiếp và tổ chức xã hội. Những bước tiến của NLP trong dịch máy, hỏi–đáp, tóm tắt, đối thoại, tìm kiếm ngữ nghĩa hay mô hình ngôn ngữ lớn đã và đang định hình lại cách chúng ta học tập, làm việc và sáng tạo.

Dẫu vậy, đằng sau sự thông minh của các hệ thống AI và NLP không có gì là quá huyền bí. Thực tế, lõi (core) của các hệ thống AI và NLP được kiến tạo, vận hành và tối ưu trên một nền tảng toán học chặt chẽ. Đại số tuyến tính cung cấp không gian biểu diễn và cơ chế tính toán: dữ liệu ngôn ngữ phi cấu trúc được số hoá thành các vector, ma trận, tensor (embeddings) và được biến đổi qua các phép toán tuyến tính trong các tầng mạng, đặc biệt trong các khối truyền thẳng và cơ chế attention. Xác suất–Thống kê thiết lập khung mô hình hoá bất định và suy luận từ dữ liệu: từ ước lượng, tổng quát hoá, đến các tiêu chí tối ưu và đánh giá dựa trên những đại lượng thông tin như entropy, cross-entropy và KL-divergence. Giải tích (Calculus) là nền tảng toán học giúp mô hình học bằng cách tự điều chỉnh tham số nhằm tối thiểu hoá sai số. Với hàm mất mát $\mathscr{L}$, giải tích đa biến lượng hoá độ nhạy theo tham số W thông qua gradient $\nabla_W \mathscr{L}$. Huấn luyện mô hình vì thế quy về bài toán tối ưu, trong đó tham số được cập nhật theo hướng ngược gradient (SGD, Momentum, Adam). Ở bình diện toàn cục, nhiều đại lượng học máy được định nghĩa như kỳ vọng trên phân phối dữ liệu, huấn luyện mini-batch là xấp xỉ Monte Carlo của các kỳ vọng này, còn trong mô hình xác suất, biên hoá (marginalization) và suy luận xấp xỉ quy về các phép tổng hợp kiểu tích phân. Vì vậy, giải tích là ngôn ngữ thống nhất của cả tối ưu hoá lẫn suy luận–tổng hợp trong các mô hình AI/NLP hiện đại.

Tuy nhiên, một nghịch lý phổ biến là chúng ta đã học rất nhiều toán ở phổ thông và đại học, nhưng sự kết nối từ định nghĩa, định lý đến ứng dụng trong khoa học kỹ thuật, đặc biệt trong công nghệ thông tin, AI và NLP, lại thường rời rạc. Chính vì vậy, Bộ sách này được tác giả biên soạn rất tâm huyết, công phu nhằm thu hẹp khoảng cách đó, giúp người học đi từ nền tảng toán học đến nguyên lý hoạt động của mô hình, từ công thức đến phép tính cụ thể, từ biết sang hiểu sâu và áp dụng.

Bộ sách Toán ứng dụng trong Trí tuệ nhân tạo và Xử lý ngôn ngữ tự nhiên là một công trình đồ sộ, lên đến gần 800 trang, được cấu trúc thành các cuốn sách khác nhau theo từng lĩnh vực hẹp của toán học, trong đó **Đại số tuyến tính và ứng dụng trong Trí tuệ nhân tạo và Xử lý ngôn ngữ tự nhiên** là cuốn sách được xuất bản đầu tiên, giúp người học dễ tiếp cận và nghiên cứu, làm nền tảng để tiếp tục hiểu sâu về tất cả các khía của toán học trong các hệ thống AI và NLP. Bạn đọc sẽ lần lượt nghiên cứu từng lát cắt nhỏ như những thành phần lõi cấu thành hệ thống AI/NLP:

từ tiền xử lý dữ liệu ngôn ngữ tự nhiên phi cấu trúc, số hóa thành các cấu trúc đại số (vector, ma trận, tensor); thực hiện các phép biến đổi và phép toán đặc trưng, sử dụng thống kê mô tả để quan sát dữ liệu, vận dụng xác suất để mô hình hóa bất định và ước lượng, ứng dụng lý thuyết thông tin để đo lường và so sánh phân phối, dùng thống kê suy luận để kiểm định và khái quát và đặc biệt vận dụng đạo hàm–tối ưu để huấn luyện mô hình. Trên nền tảng đó, Bộ sách cũng hướng dẫn xây dựng và thực hiện một quy trình (pipeline) hoàn chỉnh theo hướng dự án hệ thống AI và NLP lớn, giúp người học hợp nhất kiến thức toán vào bài toán thực tiễn.

Bộ sách được xây dựng công phu và tâm huyết từ trải nghiệm của tác giả là một người đã trải nghiệm cả hai lĩnh vực toán học và công nghệ thông tin trong suốt quá trình học phổ thông, đại học, thạc sĩ đến thời gian làm nghiên cứu sinh chuyên sâu về AI và NLP tại Đại học Bách khoa Grenoble (INPG), thực tập tiến sĩ tại Viện Tin học Quốc gia Nhật Bản (NII) và hơn 25 năm nghiên cứu, giảng dạy tại Trường Đại học Bách Khoa, Trường Đại học Công nghệ thông tin và Truyền thông Việt–Hàn (VKU), Đại học Đà Nẵng. Đồng thời, trong vai trò một nhà quản lý và lãnh đạo của trường đại học, tác giả càng thấu hiểu rằng muốn phát triển năng lực nghiên cứu, đổi mới sáng tạo bền vững, không thể thiếu một nền tảng sâu sắc về toán học và công nghệ–kỹ thuật. Với tính chất công việc quản lý rất bận rộn, nhưng bằng nỗ lực và kỷ luật cá nhân, nhiều giai đoạn tác giả thường xuyên bắt đầu quá trình biên soạn cuốn sách này từ 04 giờ sáng với sự quyết tâm cao độ để hoàn thiện bản thảo, với mong muốn đóng góp thiết thực cho cộng đồng nghiên cứu, phát triển AI/NLP, đồng thời khơi dậy niềm hứng thú của những người yêu toán học trước vẻ đẹp ứng dụng tuyệt vời mà đôi khi bị che khuất bởi sự khô khan của ký hiệu và công thức.

Trong quá trình biên soạn, tác giả đã nhận được sự hỗ trợ, giúp đỡ, góp ý của nhiều đồng nghiệp, tham khảo nhiều nguồn tài liệu quý báu và tận dụng sự hữu ích, tiên tiến của các công cụ AI ChatGPT, Google Gemini như những trợ lý ngôn ngữ và kênh tham khảo gợi ý nội dung để làm cho cuốn sách thêm phần hoàn chỉnh. Tuy nhiên, toàn bộ nội dung cuốn sách là ý tưởng tâm huyết và công sức lao động kiên trì của tác giả, do đó tác giả chịu trách nhiệm hoàn toàn về nội dung và tính đúng đắn khoa học của cuốn sách. Dù đã hết sức cẩn trọng, nội dung cuốn sách này khó tránh khỏi sai sót. Mọi góp ý xin vui lòng gửi về hcphap@vku.udn.vn để tác giả tiếp tục hoàn thiện trong các lần tái bản sau cũng như tiếp tục hoàn thiện và xuất bản các cuốn sách tiếp theo về Xác suất Thống kê, Giải tích và Tối ưu trong AL và NLP để phục vụ bạn đọc.

Huỳnh Công Pháp

GIỚI THIỆU TỔNG QUAN

Trong kiến trúc tổng thể của Trí tuệ nhân tạo (AI) và Xử lý ngôn ngữ tự nhiên (NLP) hiện đại, đại số tuyến tính là nền tảng toán học chi phối cách dữ liệu được biểu diễn, biến đổi và kết hợp để tạo ra tri thức. Nếu xem xác suất–thống kê cung cấp nguyên lý mô hình hoá sự bất định và xây dựng tiêu chuẩn đánh giá, còn giải tích cung cấp cơ chế tối ưu hoá để mô hình tự điều chỉnh tham số, thì Đại số tuyến tính (Linear Algebra) chính là ngôn ngữ cấu trúc hoá toàn bộ phép tính trong học sâu: từ biểu diễn ngôn ngữ đến cơ chế suy luận và kiến trúc mạng.

Điểm xuất phát của NLP hiện đại là việc chuyển đổi ngôn ngữ rời rạc (token, từ, câu) thành các đối tượng liên tục trong không gian nhiều chiều. Quá trình đó được thực hiện bằng embeddings với mỗi đơn vị ngôn ngữ được gán một vector $\mathbf{v} \in \mathbb{R}^d$. Từ đây, hầu hết các phép toán cốt lõi của mô hình trở thành các phép biến đổi tuyến tính và phép toán trên ma trận/tensor: các tầng tuyến tính $W\mathbf{x} + b$, chuẩn hoá, chiếu không gian, rút trích đặc trưng, và ghép nối thông tin. Trong Transformer, Attention có thể được nhìn như một chuỗi thao tác đại số tuyến tính trên các ma trận truy vấn–khóa–giá trị (Q, K, V), trong đó tích vô hướng, nhân ma trận và chuẩn hoá đóng vai trò trung tâm để lượng hoá mức độ liên quan và tổng hợp ngữ cảnh.

Trên bình diện tối ưu hoá và suy luận, đại số tuyến tính còn là nền tảng để hiểu hình học của không gian biểu diễn: chuẩn (norm) và khoảng cách để đo độ tương đồng ngữ nghĩa, trực giao và phép chiếu để tách thành phần thông tin, trị riêng và phân rã ma trận để phân tích cấu trúc (PCA, SVD), cũng như điều kiện tốt của bài toán để đảm bảo tính ổn định tính toán. Trong huấn luyện mô hình, các đối tượng gradient và cập nhật tham số cũng được biểu diễn dưới dạng vector–ma trận, cho phép triển khai hiệu quả trên phần cứng song song (GPU/TPU).

Vì vậy, cuốn sách Đại số tuyến tính và ứng dụng trong Trí tuệ nhân tạo và Xử lý ngôn ngữ tự nhiên sẽ hệ thống hoá các khái niệm Đại số tuyến tính cốt lõi như vector, không gian vector, tích vô hướng, chuẩn và khoảng cách, ma trận và phép biến đổi tuyến tính, hệ phương trình tuyến tính, trị riêng–vector riêng, các phép phân rã (đặc biệt SVD), cùng các đối tượng tensor và phép toán liên quan, đặc biệt là ứng dụng cụ thể trong các mô hình AI và NLP như xây dựng và phân tích embeddings, attention, giảm chiều, nén biểu diễn, đánh giá tương đồng ngữ nghĩa và thiết kế kiến trúc tính toán hiệu quả.

Cuốn sách không tập trung chi tiết và sâu về kiến thức Đại số tuyến tính mà chỉ hệ thống hóa lại kiến thức toán nền tảng và ứng dụng trong AI và NLP. Mục tiêu của cuốn sách là hướng đến việc thu hẹp khoảng cách giữa toán học trừu tượng với ứng dụng trong khoa học–kỹ thuật, đặc biệt trong AI và NLP. Từ định nghĩa và công thức, người học sẽ được trải nghiệm thú vị việc ứng dụng Đại số tuyến tính để xây dựng hệ thống AI và NLP thực tế với đầy đủ các bước của quy trình từ xây dựng dữ liệu, tiền xử lý, vector hóa dữ liệu, xây dựng mô hình, cài đặt thuật toán, chạy thử

nghiệm, trực quan hóa kết quả và phân tích chỉ số đánh giá. Tác giả kỳ vọng rằng, thông qua trải nghiệm này, bạn đọc sẽ nhận thức rõ lớp toán học nền tảng nằm bên dưới mỗi mô hình và thuật toán AI/NLP, từ đó có nền móng vững chắc để tiếp tục nghiên cứu, phát triển và đổi mới sáng tạo một cách bền vững.

MỤC LỤC

CHƯƠNG 1. ĐẠI SỐ TUYẾN TÍNH VÀ BIỂU DIỄN TRONG AI VÀ NLP

Giới thiệu . . . 10
1. Kỹ thuật đặc trưng . . . 11
2. Vector . . . 16
3. Ma trận . . . 17
4. Các phép toán trên vector và ma trận cơ bản . . . 17
4.1. Cộng và trừ vector . . . 17
4.2. Nhân vô hướng . . . 18
4.3. Tích vô hướng . . . 19
4.4. Chuẩn . . . 20
4.5. Chuẩn hoá . . . 20
4.6. Độ tương đồng Cosine . . . 21
4.7. Phép Xoay . . . 22
4.8. Tích Hadamard . . . 23
4.9. Nhân Ma trận–Vector và Ma trận–Ma trận . . . 23
4.10. Ma trận Chuyển vị . . . 24
4.11. Ma trận Đơn vị và Nghịch đảo . . . 25
5. Ma trận hiệp phương sai . . . 25
5.1. Khái niệm phương sai . . . 25
5.2. Ma trận hiệp phương sai . . . 26
6. Giá trị riêng và Vector riêng . . . 26
7. Phân rã giá trị suy biến (SVD) . . . 27
8. Tensor: Tổng quát hóa của Vector và Ma trận . . . 28
8.1. Các phép toán cơ bản trên Tensor . . . 29
8.2. Nhân Tensor–Ma trận (Mode-n Product) . . . 30
8.3. Tích ngoài . . . 30
8.4. Phép co Tensor . . . 31
8.5. Hoán vị và Chuyển vị Tensor . . . 32
8.6. Làm phẳng . . . 33
8.7. Tích chập trên Tensor . . . 35
Kết luận . . . 37

CHƯƠNG 2. CÁC BÀI TOÁN ĐẠI SỐ TUYẾN TÍNH TRONG AI VÀ NLP

Giới thiệu . . . 39
1. Bài toán về vector hóa dữ liệu ngôn ngữ tự nhiên . . . 39
1.1. Bài tập 1. Biểu diễn One-hot encoding và ma trận câu . . . 39
1.2. Bài tập 2. Bag-of-Words cho tập văn bản nhỏ . . . 40
1.3. Bài tập 3. TF–IDF và tầm quan trọng từ . . . 42
1.4. Bài tập 4. Embedding lookup và ma trận câu . . . 43
1.5. Bài tập 5. So sánh BoW và embedding qua độ tương đồng . . . 44
2. Các phép toán trên Vector và Ma trận . . . 46
2.1. Bài tập 6. Tích vô hướng và Cosine cơ bản . . . 46
2.2. Bài tập 7. Nhân vô hướng và co giãn . . . 48

2.3. Bài tập 8. Chuẩn hóa và Cosine 50
2.4. Bài tập 9. Nhân ma trận–vector và biến đổi tuyến tính 51
2.5. Bài tập 10. Nhân hai ma trận 53
3. Phương sai, hiệp phương sai, PCA 54
3.1. Bài tập 11. Phương sai trong AI và NLP 54
3.2. Bài tập 12. Hiệp phương sai giữa hai từ 56
3.3. Bài tập 13. Ma trận hiệp phương sai 59
4. Giá trị riêng, Vector riêng, SVD, LSA 62
4.1. Bài tập 14. Giá trị riêng và vector riêng 62
4.2. Bài tập 15. Phân rã giá trị suy biến (SVD) 65
4.3. Bài tập 16. PCA/LSA thu nhỏ với TruncatedSVD 68
5. Các phép toán với Tensor, Convolution, Attention 72
5.1. Bài tập 17. Hoán vị trục trong Multi-Head Attention 72
5.2. Bài tập 18. Reshape để gộp head trong Multi-Head Attention 74
5.3. Bài tập 19. Tích chập 1D trong TextCNN 75
5.4. Bài tập 20. TextCNN thu nhỏ: tích chập và max-pooling 78
Kết luận 81

CHƯƠNG 3. XÂY DỰNG CÁC HỆ THỐNG AI VÀ NLP

Giới thiệu 83
1. Xây dựng hệ thống AI tìm kiếm tài liệu tương đồng 83
1.1. Đề bài mô tả yêu cầu 83
1.2. Quy trình tổng quát 84
Bước 1: Tiền xử lý và xây dựng từ vựng 84
Bước 2: Tính TF, DF, IDF và TF–IDF 85
Bước 3: Biểu diễn mỗi tài liệu dưới dạng vector 88
Bước 4: Tính Cosine Similarity và giải thích 90
Bước 5: Xếp hạng và lựa chọn Top-k 91
1.3. Trực quan hoá trong không gian vector 93
1.4. Kết luận và mở rộng bài toán 94
1.5. Mã Python (kèm chú thích chi tiết) 94
2. Xây dựng hệ thống AI phân cụm văn bản 100
2.1. Đề bài mô tả yêu cầu 100
2.2. Quy trình tổng quát 100
Bước 1: Tiền xử lý văn bản 100
Bước 2: Word2Vec – Biểu diễn từ dưới dạng vector 101
Bước 3: Giảm chiều bằng SVD 105
Bước 4: Phân cụm bằng K-Means 107
2.3. Trực quan hoá phân cụm 2D 113
2.4. Kết luận và mở rộng bài toán 114
2.5. Mã Python (kèm chú thích chi tiết) 114
3. Xây dựng hệ thống AI xác định và nhóm tài liệu theo chủ đề 119
3.1. Đề bài mô tả yêu cầu 119
3.2. Quy trình tổng quát 119
Bước 1: Xây dựng ma trận TF–IDF 121
Bước 2: Phân rã SVD 122
Bước 3: Trích xuất và xác định chủ đề 123
Bước 4: Liên kết tài liệu với chủ đề 125
Bước 5: Minh họa chủ đề và Không gian ngữ nghĩa 126
3.3. Kết luận và mở rộng bài toán 127
3.4. Mã Python (kèm chú thích chi tiết) 128

Kết luận 135

CHƯƠNG 4. HỆ THỐNG PHÂN TÍCH CẢM XÚC DỮ LIỆU MẠNG XÃ HỘI BẰNG LOGISTIC REGRESSION

Giới thiệu 137
1. Đề bài mô tả yêu cầu 137
2. Quy trình tổng quát 138
3. Tiền xử lý dữ liệu văn bản mạng xã hội 139
4. Biểu diễn văn bản bằng TF–IDF và ngôn ngữ đại số tuyến tính 141
5. Logistic Regression: mô hình, hàm mất mát và tối ưu 145
6. Độ đo đánh giá mô hình phân loại 150
7. Thực nghiệm, huấn luyện mô hình, đánh giá và trực quan hoá 154
Kết luận 168

CHƯƠNG 5. BÀI TẬP NÂNG CAO VỀ ĐẠI SỐ TUYẾN TÍNH VÀ ỨNG DỤNG TRONG AI VÀ NLP

Giới thiệu 172
1. Hình học không gian nhúng và phép toán vector 172
1.1. Bài tập 1. Số học vector và bài toán tương tự 172
1.2. Bài tập 2. Nhúng câu: trung bình, dịch chuyển và vai trò độ lệch 173
1.3. Bài tập 3. Chiếu lên không gian con chủ đề bằng bình phương tối thiểu . . . 173
1.4. Bài tập 4. Trực chuẩn hóa hướng đặc trưng bằng Gram–Schmidt 174
1.5. Bài tập 5. Ma trận quay và nhúng vị trí quay (RoPE) 174
1.6. Bài tập 6. Tích Hadamard và cơ chế cổng trong GRU 175
2. Biến đổi tuyến tính và tính ổn định số trong học máy 175
2.1. Bài tập 7. Biến đổi afin như nhân ma trận mở rộng 175
2.2. Bài tập 8. Kết nối dư, ma trận đơn vị và phổ trị riêng 176
2.3. Bài tập 9. Thăm dò tuyến tính dưới dạng bài toán bình phương tối thiểu . . . 176
2.4. Bài tập 10. Hồi quy tuyến tính Ridge và nghịch đảo ổn định 177
2.5. Bài tập 11. Tự chú ý dưới dạng ma trận: chuyển vị, mặt nạ và độ phức tạp . 177
2.6. Bài tập 12. Số điều kiện và chiến lược xử lý kém điều kiện 178
3. Hiệp phương sai, phân rã riêng và làm trắng đặc trưng 178
3.1. Bài tập 13. Ma trận hiệp phương sai của embedding câu và ý nghĩa 178
3.2. Bài tập 14. Whitening embedding bằng phân rã trị riêng 179
3.3. Bài tập 15. Spectral clustering trên đồ thị đồng xuất hiện từ 179
3.4. Bài tập 16. Giảm anisotropy bằng cách loại bỏ các hướng chính 179
3.5. Bài tập 17. Phương pháp lặp luỹ thừa để xấp xỉ trị riêng lớn nhất 180
3.6. Bài tập 18. Ổn định của RNN tuyến tính hóa qua trị riêng 180
4. SVD và xấp xỉ hạng thấp để nén và căn chỉnh mô hình 181
4.1. Bài tập 19. Xấp xỉ hạng thấp cho ma trận embedding từ 181
4.2. Bài tập 20. Căn chỉnh embedding song ngữ bằng Procrustes trực giao 181
4.3. Bài tập 21. Phân rã hạng thấp cho lớp chiếu ra từ vựng 182
4.4. Bài tập 22. Giả nghịch đảo Moore–Penrose và nghiệm chuẩn nhỏ nhất . . . 182
4.5. Bài tập 23. Chuẩn Frobenius, chuẩn hạt nhân và liên hệ với hạng 183
4.6. Bài tập 24. LoRA như cập nhật hạng thấp và liên hệ với SVD 183
5. Tensor và tích chập nâng cao trong kiến trúc hiện đại 184
5.1. Bài tập 25. Tích ngoài và tương tác bậc hai trong mô hình hỏi–đáp 184

5.2. Bài tập 26. Co tensor cho Multi-Head Attention và einsum 184
5.3. Bài tập 27. Nhân tensor theo mode-n cho lớp tuyến tính trên mọi token . . . 185
5.4. Flatten/reshape giữa CNN và MLP: kiểm soát kích thước 185
5.5. Tích chập nhân quả và giãn cách: trường tiếp nhận trên chuỗi 186
5.6. Tích chập tách theo chiều sâu và đếm tham số 186
Kết luận 187

CHƯƠNG 6. THỰC HÀNH CÁC DỰ ÁN AI VÀ NLP THỰC TẾ

ỨNG DỤNG ĐẠI SỐ TUYẾN TÍNH
Giới thiệu 190
1. Xây dựng hệ thống nhận dạng thực thể và ẩn danh hoá thông tin 190
1.1. Đề bài mô tả yêu cầu 190
1.2. Quy trình tổng quát 190
Bước 1: Chuẩn hoá dữ liệu và quy ước gán nhãn 191
Bước 2: Thiết kế mô hình và đường cơ sở 192
Bước 3: Huấn luyện, điều chỉnh và kiểm soát lỗi 192
Bước 4: Ẩn danh hoá và hậu xử lý 193
Bước 5: Đánh giá và phân tích sai lỗi 193
2. Xây dựng hệ thống hỏi–đáp theo ngữ cảnh 194
2.1. Đề bài mô tả yêu cầu 194
2.2. Quy trình tổng quát 194
Bước 1: Chuẩn hoá và tổ chức dữ liệu hỏi–đáp 194
Bước 2: Tách mảnh và ánh xạ vị trí đáp án 195
Bước 3: Lựa chọn mô hình và chiến lược huấn luyện 195
Bước 4: Đánh giá và phân tích năng lực suy luận 196
Bước 5: Mở rộng theo hướng ứng dụng 196
3. Xây dựng hệ thống tóm tắt văn bản tiếng Việt 196
3.1. Đề bài mô tả yêu cầu 196
3.2. Quy trình tổng quát 196
Bước 1: Xây dựng tiêu chuẩn tóm tắt và dữ liệu 197
Bước 2: Đường cơ sở và mô hình khuyến nghị 197
Bước 3: Huấn luyện và kiểm soát hiện tượng bịa nội dung 198
Bước 4: Đánh giá và phân tích chất lượng tóm tắt 198
Bước 5: Ứng dụng và mở rộng 198
4. Xây dựng hệ thống phát hiện nội dung độc hại và thù ghét 199
4.1. Đề bài mô tả yêu cầu 199
4.2. Quy trình tổng quát 199
Bước 1: Tiền xử lý đặc thù dữ liệu mạng xã hội 199
Bước 2: Đường cơ sở và mô hình khuyến nghị 200
Bước 3: Thiết lập đánh giá đúng với bài toán mất cân bằng 200
Bước 4: Giảm thiên lệch và tăng tính an toàn khi triển khai 200
Bước 5: Tích hợp vòng lặp con người và báo cáo vận hành 201
5. Xây dựng hệ thống RAG: truy hồi ngữ nghĩa và trả lời có dẫn chứng . . . 201
5.1. Đề bài mô tả yêu cầu 201
5.2. Quy trình tổng quát 201
Bước 1: Xây dựng kho tri thức và chiến lược chia đoạn 202
Bước 2: Xây dựng bộ truy hồi và đường cơ sở truy hồi 202
Bước 3: Xếp hạng lại và lựa chọn bằng chứng 202
Bước 4: Tạo sinh câu trả lời có trích dẫn và kiểm soát bịa 203
Bước 5: Đánh giá toàn hệ thống và kịch bản triển khai 203
Kết luận 203

Chương 1

ĐẠI SỐ TUYẾN TÍNH VÀ BIỂU DIỄN DỮ LIỆU
TRONG TRÍ TUỆ NHÂN TẠO VÀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Giới thiệu 10
1 Kỹ thuật đặc trưng 11
2 Vector 16
3 Ma trận (Matrix) 17
4 Các phép toán trên vector và ma trận cơ bản 17
4.1. Cộng và trừ vector (Phép tịnh tiến – Translation) 17
4.2. Nhân vô hướng (Scaling - Co giãn) 18
4.3. Tích vô hướng (Dot Product) 19
4.4. Chuẩn (Norm) 20
4.5. Chuẩn hoá (Normalization) 20
4.6. Độ tương đồng Cosine (Cosine Similarity) 21
4.7. Phép Xoay (Rotation) 22
4.8. Tích Hadamard (Hadamard Product/Element-wise Product) 23
4.9. Nhân Ma trận–Vector và Ma trận–Ma trận 23
4.10. Ma trận Chuyển vị (Transpose) 24
4.11. Ma trận Đơn vị (Identity) và Nghịch đảo (Inverse) 25
5 Ma trận hiệp phương sai (Covariance Matrix) 25
5.1. Khái niệm phương sai (Variance): 25
5.2. Ma trận hiệp phương sai 26
6 Giá trị riêng (Eigenvalue) và Vector riêng (Eigenvector): 26
7 Phân rã giá trị suy biến (Singular Value Decomposition - SVD): . . 27
8 Tensor: Tổng quát hóa của Vector và Ma trận 28
8.1. Các phép toán cơ bản trên Tensor 29
8.2. Nhân Tensor–Ma trận (Mode-n Product) 30
8.3. Tích ngoài (Outer Product) 30
8.4. Phép co Tensor (Tensor Contraction) 31
8.5. Hoán vị và Chuyển vị Tensor (Permute/Transpose) 32
8.6. Làm phẳng (Flatten) 33
8.7. Tích chập (Convolution) trên Tensor 35
Kết luận 37

Giới thiệu

Chương này sẽ trình bày vai trò nền tảng của đại số tuyến tính trong việc biểu diễn và biến đổi dữ liệu trong các mô hình trí tuệ nhân tạo (AI) và xử lý ngôn ngữ tự nhiên (NLP). Trong các các mô hình NLP, dữ liệu huấn luyện và đầu vào tồn tại ở dạng phi cấu trúc và rời rạc (chuỗi ký tự, chuỗi token, câu, tài liệu,...). Máy tính không thao tác trực tiếp trên các đối tượng này như một thực thể mang ý nghĩa toán học, thay vào đó, toàn bộ quá trình học và suy luận của mô hình được thực hiện trên các cấu trúc số có tổ chức. Do đó, một yêu cầu cơ bản của mọi hệ thống NLP là chuyển đổi văn bản thành biểu diễn số phù hợp, đồng thời bảo toàn được thông tin cần thiết cho tác vụ.

Nội dung cốt lõi của việc chuyển đổi này thường được gọi là kỹ thuật đặc trưng hay vector hoá. Mục tiêu là ánh xạ mỗi đơn vị ngôn ngữ (từ, câu, tài liệu) vào không gian số dưới dạng vector, tập nhiều đơn vị ngôn ngữ được biểu diễn bằng ma trận, và khi xét theo lô dữ liệu, nhiều kênh đặc trưng, hoặc nhiều mức cấu trúc (ví dụ: batch, độ dài chuỗi, chiều embedding), dữ liệu được biểu diễn tự nhiên dưới dạng tensor. Ba đối tượng vector–ma trận–tensor tạo nên ngôn ngữ chung để mô tả dữ liệu, tham số mô hình, cũng như các phép biến đổi trong học sâu.

Về phương pháp, có hai hướng biểu diễn phổ biến. Nhóm biểu diễn dựa trên tần suất như Bag-of-Words và TF–IDF xem văn bản như một tập token và lượng hoá bằng thống kê đếm. Chúng có ưu điểm đơn giản, giải thích được và là nền tảng cho nhiều mô hình tuyến tính/xác suất. Nhóm biểu diễn dựa trên ngữ cảnh như Word2Vec, GloVe và các embedding theo Transformer tạo ra các vector đặc có số chiều vừa phải, cho phép mô hình hoá quan hệ ngữ nghĩa trong không gian liên tục và phụ thuộc ngữ cảnh. Dù theo hướng nào, kết quả cuối cùng vẫn là các đối tượng đại số tuyến tính, vì quá trình huấn luyện và suy luận đều quy về các phép nhân tuyến tính, tích vô hướng, chuẩn hoá, phân rã ma trận và các phép co tensor.

Để hiểu rõ mối liên hệ giữa dữ liệu ngôn ngữ tự nhiên và biểu diễn số, phần đầu của chương sẽ trình bày về kỹ thuật đặc trưng và một ví dụ minh hoạ quy trình biến đổi dữ liệu có nhãn thành các thống kê token phục vụ phân loại: từ câu đã gán nhãn, xây dựng ma trận bag-of-words, trích xuất tần suất theo lớp, tính tần suất tổng, và cuối cùng lượng hoá thiên hướng của token bằng log-odds. Ví dụ này cho thấy một điểm then chốt của NLP theo hướng thống kê: tri thức về ngôn ngữ không được giả định sẵn mà được rút ra từ dữ liệu thông qua các đại lượng có thể tính toán và kiểm chứng.

Trên nền tảng đó, phần còn lại của chương sẽ hệ thống hoá các khái niệm và phép toán cốt lõi của đại số tuyến tính và ứng dụng trong AI và NLP. Trước hết là vector và các phép toán cơ bản (cộng, nhân vô hướng, tích vô hướng, chuẩn hoá) nhằm hình thành trực giác về độ tương đồng và hình học của không gian biểu diễn. Tiếp theo là ma trận và các phép biến đổi tuyến tính, bao gồm chuyển vị, nghịch đảo, ma trận hiệp phương sai, cùng các khái niệm giá trị riêng–vector riêng và phân rã

SVD, vốn là cơ sở cho giảm chiều và mô hình hoá cấu trúc ẩn. Cuối cùng là tensor và các phép toán đa tuyến tính (permute/transpose, reshape/flatten, co tensor), đóng vai trò trung tâm trong triển khai các kiến trúc học sâu hiện đại như CNN, RNN và Transformer.

Mục tiêu của chương cung cấp định nghĩa, công thức và tổng quát hoá liên hệ giữa đại số tuyến tính với các mô hình AI/NLP: mọi mô hình NLP hiện đại đều có thể được nhìn như sự kết hợp có cấu trúc của các phép biến đổi tuyến tính và phi tuyến trong không gian vector nhiều chiều. Nắm vững đại số tuyến tính vì vậy là điều kiện cần để hiểu đúng bản chất của biểu diễn (representation), cơ chế học (learning) và suy luận (inference) trong AI và NLP.

1 Kỹ thuật đặc trưng

Trước khi biến văn bản thành số, chúng ta cần hiểu số ở đây là gì. Trong AI và NLP, các cấu trúc toán học như vector, ma trận, và tensor là nền tảng để biểu diễn và xử lý dữ liệu.

Máy tính không thể hiểu được văn bản thô (raw text), một chuỗi ký tự như "Giảng viên VKU" không mang ý nghĩa toán học. Để máy tính có thể xử lý, phân tích, hay dự đoán dựa trên văn bản, chúng ta phải thực hiện một bước gọi là kỹ thuật đặc trưng (feature Engineering) hay cụ thể hơn là vector hóa (vectorization).

Kỹ thuật đặc trưng là quá trình biến đổi mỗi đơn vị văn bản (từ, câu, hoặc tài liệu) thành một danh sách các con số, tức là một vector. Vector này được gọi là biểu diễn đặc trưng (feature representation) của văn bản.

Có hai hướng tiếp cận chính để tạo ra các vector này:

1. Biểu diễn dựa trên tần suất (count-based / sparse representation): Đây là phương pháp cổ điển, xem văn bản như một túi chứa các từ, không quan tâm đến thứ tự.

    - Bag-of-Words (BoW): Tạo một vector có số chiều bằng kích thước từ vựng ($|V|$), mỗi chiều i của vector biểu diễn cho từ thứ i trong từ vựng. Giá trị tại chiều đó là số lần từ i xuất hiện trong văn bản.
    - TF-IDF (Term Frequency-Inverse Document Frequency): Tương tự BoW, nhưng giá trị được thay bằng một trọng số, đánh giá mức độ quan trọng của từ trong văn bản đó so với toàn bộ tập dữ liệu.

    Các vector này thường có số chiều rất lớn (ví dụ: 50.000 chiều) và rất thưa (sparse), tức là hầu hết các giá trị bằng 0.

2. Biểu diễn dựa trên ngữ cảnh (Context-based / Dense Representation): Đây là phương pháp hiện đại (còn gọi là *word embedding*), học biểu diễn của từ dựa trên các từ xung quanh nó.

- Word2Vec, GloVe: Tạo ra các vector đặc (dense), tức hầu hết các giá trị đều khác 0, với số chiều nhỏ hơn nhiều (ví dụ: 300 chiều). Các vector này nắm bắt được ngữ nghĩa của từ, những từ có nghĩa tương đồng (như *vua* và *nữ hoàng*) sẽ có vector nằm gần nhau trong không gian.
- BERT, Transformer Embedding: Các mô hình này tạo ra vector biểu diễn không chỉ cho từ, mà cho cả câu, và chúng nhạy cảm với ngữ cảnh (context-sensitive). Chẳng hạn, từ *bank* trong *river bank* (bờ sông) sẽ có vector khác với *bank* trong *bank account* (tài khoản ngân hàng).

Cho dù bất kể sử dụng phương pháp nào (BoW, TF-IDF, hay Word2Vec...), kết quả cuối cùng luôn là dữ liệu có cấu trúc. Một từ hoặc tài liệu được biểu diễn bằng một vector. Một tập tài liệu (corpus) hoặc một câu (gồm nhiều vector từ) được biểu diễn bằng một ma trận và một *batch* dữ liệu (gồm nhiều câu) để huấn luyện mô hình học sâu sẽ được biểu diễn bằng một tensor.

Do đó, chúng ta cần phải hiểu rõ các cấu trúc toán học và nắm rõ kiến thức đại số tuyến tính.

Sau đây là một ví dụ minh họa kỹ thuật đặc trưng để hiểu rõ mối liên hệ giữa dữ liệu ngôn ngữ tự nhiên và biểu diễn số:

Từ các câu đã được gán nhãn của tập huấn luyện đầu vào, sau đây là quy trình minh họa cách để biến đổi các câu thành ma trận số:

$$\text{Câu có nhãn} \to \text{Bag-of-Words} \to \text{Tần suất theo lớp} \to$$
$$\to \text{Tần suất tổng} \to \text{Log-odds theo token}$$

Bước 1: Giả định dữ liệu huấn luyện

Xét tập dữ liệu huấn luyện gồm N câu (tweet), mỗi câu đã được gán nhãn cảm xúc:

$$\mathscr{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N},$$

trong đó $x^{(i)}$ là câu thứ i (chuỗi token sau tiền xử lý) và $y^{(i)} \in \{0,1\}$ là nhãn cảm xúc (0: tiêu cực, 1: tích cực). Biểu diễn $\mathscr{D}$ cho thấy đây là bài toán học có giám sát: nhãn y cung cấp thông tin để lượng hoá liên hệ giữa token và lớp. Nếu không có nhãn, chỉ có thể thống kê mức độ xuất hiện chung của token, chúng ta không thể suy luận token nghiêng về cảm xúc nào.

Ví dụ dữ liệu các câu huấn luyện đã được gán nhãn:

Nhãn y	Câu x
1	“I am very happi today”
0	“This movie is bad and sad”
1	“What a great and happi day”
0	“I feel sad and bad”

Bước 2: Tiền xử lý và tập từ vựng

Mỗi câu được tiền xử lý theo chuỗi thao tác phổ biến gồm chuyển về chữ thường, tách token, loại bỏ dấu câu và chuẩn hoá dạng từ bằng stemming/lemmatization (ví dụ "happy" $\to$ "happi"). Tiền xử lý làm giảm nhiễu và gom các biến thể hình thái về một dạng chuẩn, nhờ đó giảm kích thước từ vựng, tăng số lần quan sát hữu ích cho mỗi token và cải thiện độ ổn định của thống kê. Sau tiền xử lý, ta thu được tập từ vựng:

$$\mathscr{V} = \{\text{happi}, \text{sad}, \text{great}, \text{bad}, \ldots\}.$$

Tập từ vựng $\mathscr{V}$ là không gian đặc trưng rời rạc dùng để biểu diễn văn bản thành dữ liệu số.

Bước 3: Biểu diễn bag-of-words

Mỗi câu $x^{(i)}$ được biểu diễn dưới dạng vector đếm:

$$\mathbf{x}^{(i)} = (c_1^{(i)}, c_2^{(i)}, \ldots, c_{|\mathscr{V}|}^{(i)}),$$

trong đó $c_j^{(i)}$ là số lần token $w_j \in \mathscr{V}$ xuất hiện trong câu i. Bag-of-words giả định rằng thông tin phân loại chủ yếu đến từ sự xuất hiện và tần suất của token, không xét thứ tự từ. Biểu diễn này cho phép trích xuất trực tiếp các thống kê dựa trên đồng hiện token–nhãn, đồng thời cung cấp dữ liệu chuẩn cho các mô hình tuyến tính và xác suất.

Để đơn giản cho minh họa tính toán, chúng ta chỉ xét một số từ vựng rút gọn $\mathscr{V} = \{\text{happi}, \text{sad}, \text{great}, \text{bad}\}$, ta có:

Câu	happi	sad	great	bad	y
I am very happi today	1	0	0	0	1
This movie is bad and sad	0	1	0	1	0
What a great and happi day	1	0	1	0	1
I feel sad and bad	0	1	0	1	0

Bảng trên là một ma trận bag-of-words có nhãn. Mỗi cột tương ứng một token, mỗi hàng tương ứng một câu/mẫu, và nhãn y gắn với từng câu/mẫu để từ đây chúng ta có thể vừa gom thống kê theo lớp, vừa tính được mức độ xuất hiện chung của token trên toàn bộ tập dữ liệu.

Bước 4: Xây dựng bảng tần suất theo lớp

Từ dữ liệu đã gán nhãn, ta định nghĩa:

$$\text{freqs}(w, 1) = \sum_{i=1}^{N} c_w^{(i)} \cdot \mathbb{I}[y^{(i)} = 1], \tag{1}$$

$$\text{freqs}(w, 0) = \sum_{i=1}^{N} c_w^{(i)} \cdot \mathbb{I}[y^{(i)} = 0], \tag{2}$$

trong đó,

- $c_w^{(i)}$ là số lần token w xuất hiện trong câu i.
- $\mathbb{I}[\cdot]$ là hàm chỉ thị.
- $\text{freqs}(w,1)$ và $\text{freqs}(w,0)$ là thống kê có điều kiện theo lớp, cho biết token w xuất hiện bao nhiêu lần trong từng lớp, làm cơ sở để nhận định token nghiêng về lớp nào và là đầu vào cho các thước đo phân biệt lớp.

Kết quả tần suất theo lớp.

Token w	$\text{freqs}(w,1)$	$\text{freqs}(w,0)$
happi	2	0
great	1	0
sad	0	2
bad	0	2

Bước 5: Tần suất tổng và trực giác về mức độ quan trọng

Từ các tần suất theo lớp, ta định nghĩa thêm tần suất tổng của token:

$$\text{freq}(w) = \text{freqs}(w,1) + \text{freqs}(w,0).$$

$\text{freq}(w)$ đo mức độ phổ biến của token trên toàn bộ tập huấn luyện, hữu ích cho mô tả dữ liệu và tiền xử lý: lọc token quá hiếm để giảm nhiễu và giảm độ thưa, đặt ngưỡng xuất hiện tối thiểu khi xây dựng từ vựng và đánh giá sơ bộ độ ổn định của các ước lượng theo lớp.

Token w	$\text{freqs}(w,1)$	$\text{freqs}(w,0)$	$\text{freq}(w)$
happi	2	0	2
great	1	0	1
sad	0	2	2
bad	0	2	2

Bảng 1: Tần suất theo lớp và tần suất tổng của token

Mặc dù phản ánh độ phổ biến, $\text{freq}(w)$ không thể hiện hướng phân biệt lớp vì không phân tách theo nhãn. Một token có thể có $\text{freq}(w)$ lớn nhưng xuất hiện gần như cân bằng ở hai lớp nên ít giá trị phân loại; ngược lại, hai token có cùng $\text{freq}(w)$ nhưng thiên hướng đối lập, chẳng hạn $(50,0)$ và $(0,50)$ đều cho $\text{freq}(w) = 50$ nhưng ý nghĩa phân loại trái ngược. Trong phân loại, điều cần lượng hoá không phải mức độ xuất hiện tuyệt đối mà là mức độ chênh lệch giữa hai lớp. Vì vậy thước đo thiên hướng cần phụ thuộc đồng thời vào $\text{freqs}(w,1)$ và $\text{freqs}(w,0)$ theo dạng so sánh, có dấu để biểu thị hướng và ổn định khi tần suất nhỏ hoặc khi một phía bằng 0. Các

yêu cầu này dẫn đến log-odds theo token ở bước tiếp theo. Đồng thời, $\text{freq}(w)$ vẫn hữu ích như một chỉ báo về độ tin cậy: token có $\text{freq}(w)$ nhỏ thường cho ước lượng log-odds kém ổn định và có thể cần lọc hoặc làm trơn mạnh hơn.

Bước 6: Tính log-odds từ dữ liệu đã gán nhãn

Với mỗi token w, ta tính:

$$s(w) = \log\left(\frac{\text{freqs}(w,1)+1}{\text{freqs}(w,0)+1}\right).$$

Tần suất tổng $\text{freq}(w)$ phản ánh độ phổ biến nhưng không phản ánh hướng phân biệt lớp, trong khi $s(w)$ là thước đo phân biệt lớp có dấu: $s(w) > 0$ cho thấy w xuất hiện tương đối nhiều hơn trong lớp tích cực, $s(w) < 0$ cho thấy w xuất hiện tương đối nhiều hơn trong lớp tiêu cực, và $|s(w)|$ phản ánh độ mạnh của bằng chứng mà token cung cấp. Lấy log tạo thang đo đối xứng quanh 0 và nén các tỉ lệ lớn; hằng số $+1$ là làm trơn để tránh chia cho 0 và tránh kết luận tuyệt đối khi dữ liệu hữu hạn.

Với dữ liệu ví dụ, ta có:

$$s(\text{happi}) = \log\left(\frac{2+1}{0+1}\right) = \log(3) \approx 1.0986,$$

$$s(\text{great}) = \log\left(\frac{1+1}{0+1}\right) = \log(2) \approx 0.6931,$$

$$s(\text{sad}) = \log\left(\frac{0+1}{2+1}\right) = \log\left(\frac{1}{3}\right) \approx -1.0986,$$

$$s(\text{bad}) = \log\left(\frac{0+1}{2+1}\right) = \log\left(\frac{1}{3}\right) \approx -1.0986.$$

Token w	$\text{freqs}(w,1)$	$\text{freqs}(w,0)$	$\text{freq}(w)$	$s(w)$	Thiên hướng
happi	2	0	2	$+1.0986$	tích cực
great	1	0	1	$+0.6931$	tích cực
sad	0	2	2	-1.0986	tiêu cực
bad	0	2	2	-1.0986	tiêu cực

Bảng 2: Bảng log-odds theo token từ dữ liệu huấn luyện

Bảng 2 cho thấy rõ sự khác biệt giữa ba khái niệm:

- $\text{freqs}(w,1)$ và $\text{freqs}(w,0)$ mô tả cách token phân bố theo lớp,
- $\text{freq}(w)$ mô tả mức độ phổ biến tổng thể của token,
- $s(w)$ mô tả mức độ và hướng mà token đó làm nghiêng quyết định phân loại.

Các token “happi” và “great” có $s(w) > 0$, nghĩa là khi xuất hiện, chúng làm tăng log-odds nghiêng về lớp tích cực. Ngược lại, “sad” và “bad” có $s(w) < 0$, nên đóng vai trò bằng chứng cho lớp tiêu cực. Độ lớn $|s(w)|$ phản ánh mức độ mạnh của bằng chứng: trong ví dụ này, “happi” và “sad” mang bằng chứng mạnh hơn “great” do tỉ lệ xuất hiện theo lớp chênh lệch lớn hơn.

Bảng log-odds theo token có thể được xem như một *từ điển trọng số cảm xúc* (sentiment lexicon) được học trực tiếp từ dữ liệu. Trong các mô hình xác suất như Naive Bayes, tổng các giá trị $s(w)$ của các token trong một câu xấp xỉ log-odds của nhãn câu đó. Trong các mô hình tuyến tính như Logistic Regression, $s(w)$ đóng vai trò tương tự một trọng số đặc trưng có dấu.

Quan trọng hơn, bảng này minh hoạ cách tri thức ngôn ngữ được trích xuất một cách định lượng: mỗi token không chỉ được xem là một ký hiệu rời rạc, mà trở thành một nguồn bằng chứng có cường độ và hướng xác định, từ đó cho phép mô hình NLP đưa ra quyết định phân loại một cách có cơ sở thống kê.

Từ nội dung và ví dụ minh họa về kỹ thuật đặc trưng cho phép biến đổi dữ liệu ngôn ngữ tự nhiên thành vector, ma trận số ở trên, chúng ta nhận thấy Đại số tuyến tính có vai trò rất quan trọng, là kiến thức nền tảng và công cụ cốt lõi cho phép biểu diễn (reprsentation) và biến đổi (transformation) dữ liệu các mô hình AI và NLP.

Trong NLP, văn bản thô (raw text) thực chất là dữ liệu phi cấu trúc và mang tính rời rạc, chúng sẽ được biến đổi và biểu diễn dưới dạng cấu trúc dữ liệu số như Vector (Tensor 1D), Ma trận (Tensor 2D), và Tensor (Tensor nD) để lượng hóa (quantify) và nhúng (embed) thông tin ngữ nghĩa. Thông qua các kỹ thuật như word embeddings (ví dụ: Word2Vec, GloVe), mỗi đơn vị ngôn ngữ (từ, câu) được ánh xạ vào một không gian liên tục đa chiều. Trong không gian này, các mối quan hệ ngữ nghĩa (như "king" - "man"+ "woman"$\approx$ "queen") có thể được mô hình hóa và tính toán. Trong các nội dung tiếp theo, chúng ta sẽ hệ thống hóa lại các kiến thức nền tảng của Đại số tuyến tính và ứng dụng trong các mô hình AI và NPL.

2 Vector

Trong các mô hình AI và NLP, vector thực chất là một danh sách (mảng 1 chiều) các số thực có thứ tự:

$$\vec{v} = [v_1, v_2, \ldots, v_n]$$

Ứng dụng trong mô hình NLP: Mỗi vector biểu diễn một đối tượng (ví dụ: một từ, một câu hoặc một văn bản). Số chiều n đại diện cho số lượng đặc trưng (*features*) mô tả đối tượng đó. Ví dụ, từ “VKU” có thể được biểu diễn bởi vector 300 chiều trong không gian embedding Word2Vec.

3 Ma trận (Matrix)

Ma trận là tập hợp các vector sắp xếp thành bảng 2 chiều có m hàng và n cột:

$$M = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1n} \\ m_{21} & m_{22} & \dots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{m1} & m_{m2} & \dots & m_{mn} \end{bmatrix}$$

Ứng dụng trong NLP: Một ma trận có thể biểu diễn toàn bộ tập văn bản (*corpus*):

- m (số hàng): số văn bản trong tập dữ liệu.
- n (số cột): kích thước từ vựng (*vocabulary size*), ký hiệu $|V|$.
- Mỗi hàng M_i biểu diễn vector đặc trưng của văn bản thứ i.

4 Các phép toán trên vector và ma trận cơ bản

4.1. Cộng và trừ vector (Phép tịnh tiến – Translation)

Mục đích của các phép toán này nhằm dịch chuyển một vector trong không gian bằng một vector khác.

$$\vec{u} + \vec{v} = [u_1 + v_1, u_2 + v_2, \dots]$$

Các phép toán này được áp dụng để thực hiện các phép toán tương tự (analogy) trong không gian embedding, tìm kiếm các mối quan hệ ngữ nghĩa. Phép trừ vector $\vec{u} - \vec{v}$ cho ra vector chỉ hướng và khoảng cách từ $\vec{v}$ đến $\vec{u}$.

Một ví dụ ngữ nghĩa kinh điển trong NLP:

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

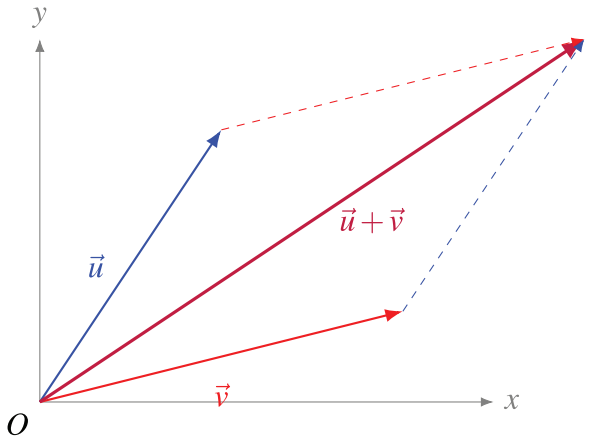
Giả sử trong không gian 3 chiều (giản lược):

$$\begin{aligned} \vec{v}_{\text{king}} &= [0.8, 0.2, 0.1], \\ \vec{v}_{\text{man}} &= [0.7, 0.1, 0.0], \\ \vec{v}_{\text{woman}} &= [0.65, 0.15, 0.05] \end{aligned}$$

Khi đó, vector kết quả (biểu diễn cho "queen") sẽ là:

$$[0.8 - 0.7 + 0.65,\ 0.2 - 0.1 + 0.15,\ 0.1 - 0.0 + 0.05] = [0.75, 0.25, 0.15]$$

Vector này sẽ nằm rất gần vector thực tế của từ "queen" trong không gian embedding.



Hình 1: Minh họa phép cộng vector (Quy tắc hình bình hành).

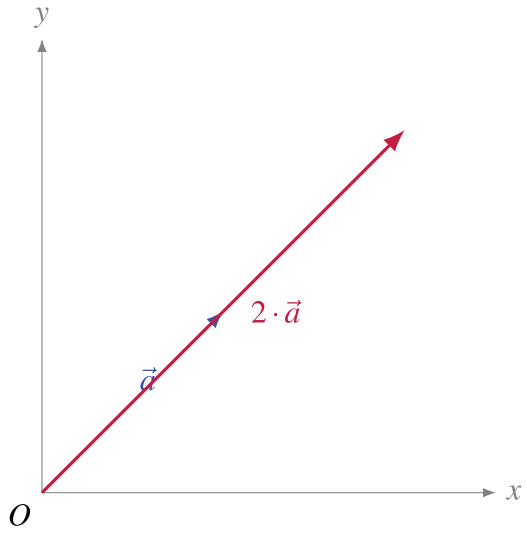
4.2. Nhân vô hướng (Scaling - Co giãn)

Phép toán này sẽ làm vector dài hoặc ngắn hơn mà không đổi hướng.

$$s \cdot \vec{a} = [sa_1, sa_2, \ldots, sa_n]$$

Phép toán này có ý nghĩa áp dụng nhằm thay đổi *cường độ* hoặc *tầm quan trọng* của một đặc trưng. Trong NLP, phép toán này dùng để chuẩn hóa các đặc trưng (features) (ví dụ: nhân với $1/\max$) hoặc có thể dùng để nhấn mạnh (up-weight) hoặc giảm nhẹ (down-weight) tầm quan trọng của một từ/tài liệu.

Ví dụ: Giả sử $\vec{v} = [0.1, 0.5]$ là vector TF-IDF của từ "VKU" trong 2 tài liệu D1 và D2. Nếu ta cho rằng D2 quan trọng gấp đôi, ta có thể áp dụng trọng số $s = 2$ cho chiều thứ hai (mặc dù phép toán này thường áp dụng cho toàn vector). Một ví dụ phổ biến hơn là trong thuật toán Gradient Descent, ta nhân vector gradient (hướng dốc nhất) với một vô hướng s (learning rate) để cập nhật trọng số. *Ví dụ:* $2 \cdot [1, 3, -2] = [2, 6, -4]$.



Hình 2: Minh họa phép nhân vô hướng (Scaling).

4.3. Tích vô hướng (Dot Product)

Đo độ tương đồng (similarity) giữa hai vector:

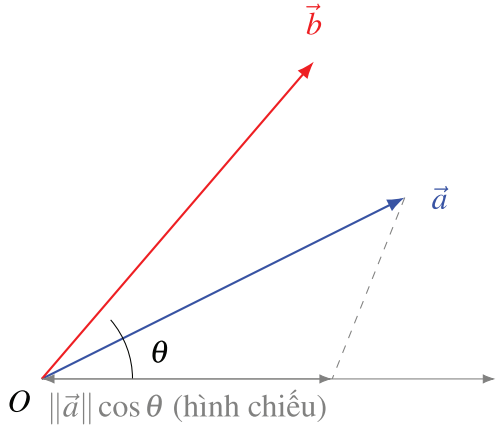
$$\vec{a} \cdot \vec{b} = \sum_i a_i b_i = \|\vec{a}\| \|\vec{b}\| \cos(\theta)$$

Phép toán này có ý nghĩa áp dụng nền tảng để tính toán độ tương đồng (cosine similarity) của 2 vector nhằm giúp so sánh hai đơn vị ngôn ngữ như 2 từ có tương đồng về ngữ nghĩa hay không.

- Kết quả dương (> 0) và càng lớn, hai vector càng cùng hướng (góc $\theta < 90°$).
- Kết quả âm (< 0) và càng nhỏ, hai vector càng ngược hướng (góc $\theta > 90°$).
- Nếu kết quả bằng 0, hai vector vuông góc (trực giao, $\theta = 90°$).

Ví dụ: Dùng trong các phép tính của mạng nơ-ron, ví dụ tính output của một nơ-ron: $y = \sum(w_i x_i) + b = \vec{w} \cdot \vec{x} + b$.

Ví dụ 1: $\vec{a} = [1,2,3], \vec{b} = [4,5,6] \Rightarrow \vec{a} \cdot \vec{b} = 32$ (cùng hướng).
Ví dụ 2: $\vec{c} = [1,2], \vec{d} = [-3,1] \Rightarrow \vec{c} \cdot \vec{d} = (1 \cdot -3) + (2 \cdot 1) = -1$ (hơi ngược hướng).



Hình 3: Minh họa Tích vô hướng (Độ tương đồng / Hình chiếu).

4.4. Chuẩn (Norm)

Đo lường *độ lớn* hoặc *chiều dài* của vector.

- L2-norm (Euclidean): $\|\vec{a}\|_2 = \sqrt{\sum a_i^2}$. Là khoảng cách Euclid từ gốc tọa độ.
- L1-norm (Manhattan): $\|\vec{a}\|_1 = \sum |a_i|$. Là tổng các giá trị tuyệt đối.

L2-norm có ý nghĩa ứng dụng dùng để đo khoảng cách (Euclidean distance) và trong phép chuẩn hóa (normalization). L1-norm thường dùng trong các mô hình cần sự thưa (sparsity), như Lasso Regression, vì nó có xu hướng đẩy các trọng số không quan trọng về 0.

4.5. Chuẩn hoá (Normalization)

Biến vector thành vector đơn vị (có độ dài L2-norm bằng 1):

$$\hat{a} = \frac{\vec{a}}{\|\vec{a}\|_2}$$

Phép toán này có ý nghĩa ứng dụng nhằm loại bỏ ảnh hưởng của độ lớn (magnitude) và chỉ giữ lại hướng (direction) của vector. Đây là bước bắt buộc khi tính Cosine Similarity, đảm bảo rằng độ tương đồng chỉ phụ thuộc vào góc θ giữa hai vector, chứ không phụ thuộc vào độ dài của chúng.

Ví dụ: Vector $\vec{v}_{\text{apple}} = [1,1,0]$ và $\vec{v}_{\text{apples}} = [2,2,0]$. Rõ ràng chúng rất giống nhau. Nếu không chuẩn hóa, tích vô hướng là 4. Chuẩn hóa: $\|\vec{v}_{\text{apple}}\|_2 = \sqrt{2}$, $\|\vec{v}_{\text{apples}}\|_2 = \sqrt{8}$. $\hat{v}_{\text{apple}} = [1/\sqrt{2}, 1/\sqrt{2}, 0] \approx [0.707, 0.707, 0]$. $\hat{v}_{\text{apples}} = [2/\sqrt{8}, 2/\sqrt{8}, 0] \approx [0.707, 0.707, 0]$. Hai vector sau khi chuẩn hóa là giống hệt nhau, cho độ tương đồng Cosine = 1.

4.6. Độ tương đồng Cosine (Cosine Similarity)

Đo lường góc θ giữa hai vector, bỏ qua độ lớn của chúng.

$$\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|_2 \|\vec{b}\|_2} = \frac{\sum_i a_i b_i}{\sqrt{\sum a_i^2}\sqrt{\sum b_i^2}}$$

Phép toán này tương đương với việc lấy tích vô hướng của hai vector đã được chuẩn hóa (normalized).

Phép toán này là thước đo tương đồng (similarity) phổ biến nhất trong NLP và các hệ thống gợi ý (Recommender Systems).

- $\cos(\theta) = 1$: Hai vector cùng hướng (hoàn toàn tương đồng).
- $\cos(\theta) = 0$: Hai vector trực giao (không liên quan).
- $\cos(\theta) = -1$: Hai vector ngược hướng (hoàn toàn trái nghĩa).

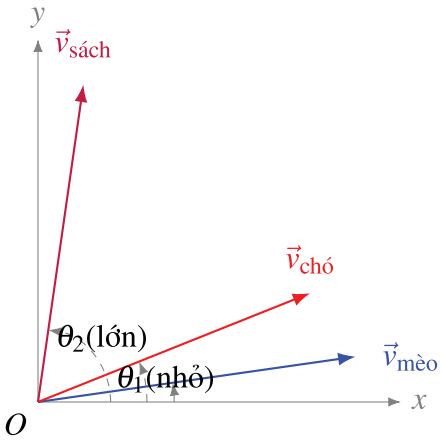
Tương đồng cosine hoạt động tốt vì trong NLP chúng ta quan tâm đến hướng ngữ nghĩa của từ chứ không phải độ lớn (ví dụ: tần suất xuất hiện của từ).

Ví dụ: Tìm độ tương đồng giữa 3 từ:
$\vec{v}_{\text{mèo}} = [0.8, 0.1]$, $\vec{v}_{\text{chó}} = [0.7, 0.2]$, $\vec{v}_{\text{sách}} = [0.1, 0.9]$.

Tích vô hướng (tử số):
$\vec{v}_{\text{mèo}} \cdot \vec{v}_{\text{chó}} = (0.8 \cdot 0.7) + (0.1 \cdot 0.2) = 0.58$
$\vec{v}_{\text{mèo}} \cdot \vec{v}_{\text{sách}} = (0.8 \cdot 0.1) + (0.1 \cdot 0.9) = 0.17$

Chuẩn L2 (mẫu số):
$\|\vec{v}_{\text{mèo}}\| = \sqrt{0.8^2 + 0.1^2} \approx 0.806$
$\|\vec{v}_{\text{chó}}\| = \sqrt{0.7^2 + 0.2^2} \approx 0.728$
$\|\vec{v}_{\text{sách}}\| = \sqrt{0.1^2 + 0.9^2} \approx 0.905$

Độ tương đồng Cosine:
$\cos(\text{mèo}, \text{chó}) = \frac{0.58}{0.806 \cdot 0.728} \approx 0.988$ (Rất cao, gần 1)
$\cos(\text{mèo}, \text{sách}) = \frac{0.17}{0.806 \cdot 0.905} \approx 0.233$ (Rất thấp, gần 0)



Hình 4: Minh họa Cosine Similarity: Góc θ_1 (mèo, chó) nhỏ $\Rightarrow \cos(\theta_1) \approx 1$. Góc θ_2 (mèo, sách) lớn $\Rightarrow \cos(\theta_2) \approx 0$.

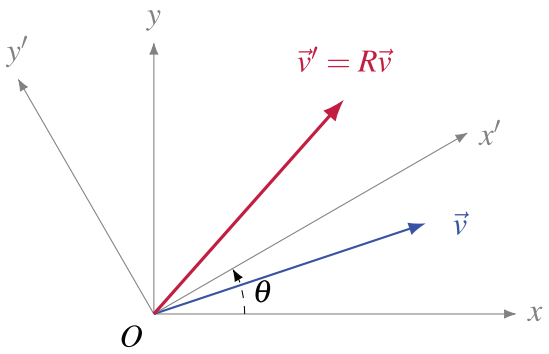
4.7. Phép Xoay (Rotation)

Sử dụng ma trận xoay (Rotation Matrix) để xoay vector một góc θ:

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \vec{v}' = R\vec{v}$$

Phép toán này có ý nghĩa ứng dụng nhằm thay đổi hệ quy chiếu (coordinate system) của dữ liệu mà không làm thay đổi mối quan hệ nội tại (khoảng cách, góc) giữa các điểm dữ liệu.

Ví dụ: Trong *PCA (Principal Component Analysis - Phân tích thành phần chính)*, chúng ta tìm một ma trận R (gồm các vector riêng - eigenvectors) để xoay toàn bộ tập dữ liệu. Mục đích là các trục tọa độ mới (principal components) sẽ thẳng hàng với các hướng có phương sai (variance) lớn nhất của dữ liệu. Điều này cho phép ta giữ lại nhiều thông tin nhất khi giảm chiều bằng cách loại bỏ các trục có phương sai thấp.



Hình 5: Minh họa Phép Xoay (Rotation).

4.8. Tích Hadamard (Hadamard Product/Element-wise Product)

Nhân từng phần tử tương ứng của hai vector (hoặc ma trận) cùng kích thước:

$$\vec{a} \odot \vec{b} = [a_1 b_1, a_2 b_2, \dots]$$

Phép toán này có ý nghĩa ứng dụng dùng trong các cơ chế cổng (gating mechanisms) của mạng nơ-ron. Một vector (thường chứa giá trị từ 0 đến 1, qua hàm sigmoid) hoạt động như một mặt nạ (mask) hoặc cổng (gate), quyết định bao nhiêu thông tin từ vector kia được phép đi qua tại mỗi chiều.

Ví dụ: Trong mạng LSTM(Long Short-Term Memory), cổng quên (forget gate) $\vec{f}_t$ và cổng đầu vào (input gate) $\vec{i}_t$ được nhân Hadamard với ô trạng thái (cell state) cũ $\vec{C}_{t-1}$ và cell state ứng cử $\vec{\tilde{C}}_t$ để tính cell state mới:

$$\vec{C}_t = (\vec{f}_t \odot \vec{C}_{t-1}) + (\vec{i}_t \odot \vec{\tilde{C}}_t)$$

Nếu $f_{t,j} = 0$, thông tin ở chiều j của $\vec{C}_{t-1}$ sẽ bị lờ đi (bỏ qua). Nếu $i_{t,j} = 1$, toàn bộ thông tin mới ở chiều j của $\vec{\tilde{C}}_t$ sẽ được thêm vào.

4.9. Nhân Ma trận–Vector và Ma trận–Ma trận

Là phép biến đổi tuyến tính (linear transformation), ánh xạ dữ liệu từ không gian n chiều sang không gian m chiều.

$$(M\vec{v})_i = \sum_j M_{ij} v_j \quad \text{hoặc} \quad C_{ik} = \sum_j A_{ij} B_{jk}$$

Đây là phép toán nền tảng của các lớp (layers) trong mạng nơ-ron (ví dụ: Fully-Connected Layer, Lớp Q, K, V trong Transformer). Ma trận M chính là tập hợp các trọng số (weights) mà mô hình học được.

Ví dụ: Ví dụ trong Self-Attention (mô hình Transformer): Cho ma trận X (2 từ $\times$ 3 chiều embedding): $X = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$.

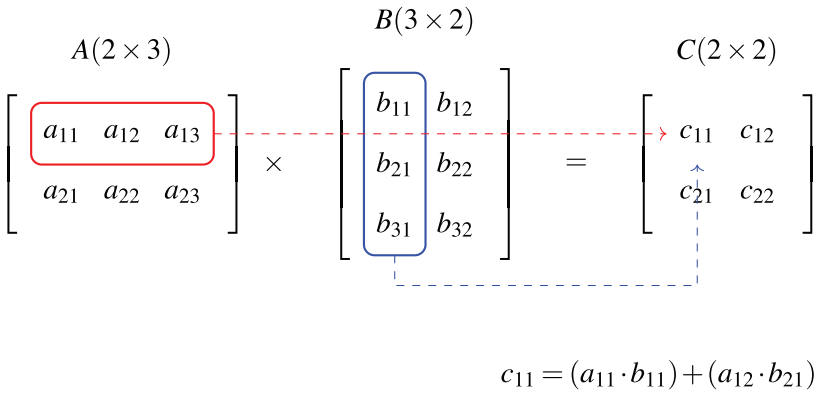
Ma trận trọng số W_Q (3 chiều $\times$ 2 chiều) $W_Q = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$.

Ma trận Query $Q = XW_Q = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$.

Tương tự, ta tính K và V. Phép nhân QK^T (nhân ma trận-ma trận) sẽ tính toán ma trận Attention Score, cho biết mức độ *chú ý* giữa các từ.

$$Q = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}, \quad K^T = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \Rightarrow \text{Attention} = QK^T = \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}$$

Ma trận kết quả biểu diễn mức độ *chú ý* (attention) giữa các từ (ví dụ: từ 1 chú ý đến từ 1 với điểm là 2, từ 1 chú ý đến từ 2 với điểm là 1).



Hình 6: Minh họa phép nhân ma trận: Phần tử C_{ij} là tích vô hướng của hàng i của A và cột j của B.

4.10. Ma trận Chuyển vị (Transpose)

Phép toán này sẽ chuyển đổi ma trận qua đường chéo chính, biến hàng thành cột và cột thành hàng.

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \Rightarrow \quad A^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

Kích thước $m \times n$ sẽ trở thành $n \times m$.

Đây là một trong những phép toán phổ biến nhất, dùng để thay đổi kích thước ma trận cho phù hợp với phép nhân, hoặc để tính toán ma trận hiệp phương sai.

Ví dụ:

- Trong mô hình Self-Attention: Phép tính cốt lõi là $Attention = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})$. Giả sử Q và K đều có kích thước [seq_len $\times$ dim]. Để so sánh mỗi từ với mọi từ khác, ta cần K có kích thước [dim $\times$ seq_len]. Phép K^T thực hiện điều này, cho ra ma trận QK^T kích thước [seq_len $\times$ seq_len].
- Trong Hồi quy tuyến tính (Linear Regression): Ma trận X có kích thước [samples $\times$ features]. Ma trận X^T [features $\times$ samples] được dùng để tính ma trận (X^TX).

4.11. Ma trận Đơn vị (Identity) và Nghịch đảo (Inverse)

Ma trận đơn vị I là ma trận vuông mà giá trị các phần tử trên đường chéo chính là 1 và các phần tử khác có giá trị 0. Ta có: $AI = IA = A$.

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Ma trận nghịch đảo A^{-1} (chỉ tồn tại với ma trận vuông không suy biến) là ma trận mà khi nhân với A sẽ ra ma trận đơn vị I.

$$AA^{-1} = A^{-1}A = I$$

Ma trận nghịch đảo A^{-1} tương đương với phép chia $1/A$, được dùng để giải các hệ phương trình tuyến tính. Nếu $A\vec{x} = \vec{b}$, thì $\vec{x} = A^{-1}\vec{b}$.

Ví dụ: Trong Hồi quy tuyến tính (Linear Regression), công thức tìm trọng số tối ưu $\vec{w}$ (Normal Equation) là:

$$\vec{w} = (X^T X)^{-1} X^T \vec{y}$$

Trong đó, $(X^T X)^{-1}$ chính là phép nghịch đảo ma trận $(X^T X)$, cho phép chúng ta giải $\vec{w}$ một cách trực tiếp.

5 Ma trận hiệp phương sai (Covariance Matrix)

5.1. Khái niệm phương sai (Variance):

Phương sai cho phép đo lường mức độ phân tán (spread) của một tập dữ liệu (một feature, một cột) so với giá trị trung bình (mean) μ của nó. Dùng $N-1$ cho thống kê mẫu (sample variance).

$$\sigma^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu)^2$$

Phương sai có ý nghĩa ứng dụng cho biết lượng thông tin mà một đặc trưng (feature) mang lại.

- Phương sai cao: Dữ liệu phân tán rộng, đặc trưng này có khả năng phân biệt tốt.
- Phương sai thấp (gần 0): Dữ liệu gần như là hằng số, đặc trưng này vô dụng và có thể bị loại bỏ (feature selection).

Ví dụ: Trong một khảo sát khách hàng, đặc trưng tuổi (từ 18 đến 60) sẽ có phương sai cao, trong khi đặc trưng số mắt (hầu hết là 2) sẽ có phương sai bằng 0 và không có giá trị cho mô hình dự đoán.

5.2. Ma trận hiệp phương sai

Mở rộng của phương sai cho nhiều chiều (features) là một ma trận vuông Σ kích thước $n \times n$ (với n là số features).

- Các phần tử trên đường chéo chính $\Sigma_{ii} = \sigma^2(x_i)$ là phương sai (variance) của feature i.
- Các phần tử ngoài đường chéo $\Sigma_{ij} = \text{Cov}(x_i, x_j)$ (với $i \neq j$) là Hiệp phương sai (covariance) giữa feature i và feature j.

$$\text{Cov}(X,Y) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

$$\Sigma = \begin{bmatrix} \sigma^2(x_1) & \text{Cov}(x_1,x_2) & \dots & \text{Cov}(x_1,x_n) \\ \text{Cov}(x_2,x_1) & \sigma^2(x_2) & \dots & \text{Cov}(x_2,x_n) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(x_n,x_1) & \text{Cov}(x_n,x_2) & \dots & \sigma^2(x_n) \end{bmatrix}$$

Ma trận này có vai trò tóm tắt toàn bộ cấu trúc tuyến tính và mối tương quan giữa các đặc trưng. $\text{Cov}(X,Y) > 0$ nghĩa là X tăng thì Y có xu hướng tăng; $\text{Cov}(X,Y) < 0$ nghĩa là X tăng thì Y có xu hướng giảm.
Ví dụ: Ma trận Hiệp phương sai là đầu vào cốt lõi cho thuật toán PCA, nếu ta có ma trận TF-IDF (văn bản $\times$ từ), ma trận hiệp phương sai (từ $\times$ từ) sẽ cho biết các cặp từ nào có xu hướng cùng xuất hiện (hoặc không cùng xuất hiện) trong các văn bản, dựa trên sự biến động của chúng quanh giá trị trung bình.

6 Giá trị riêng (Eigenvalue) và Vector riêng (Eigenvector):

Với một ma trận vuông A (ví dụ: ma trận hiệp phương sai), một vector riêng $\vec{v}$ là một vector đặc biệt mà khi A tác động (transform) lên nó, nó không đổi hướng, chỉ co giãn (scale) một lượng đúng bằng giá trị riêng λ tương ứng.

$$A\vec{v} = \lambda\vec{v}$$

Phép phân rã (decomposition) ma trận A thành các vector riêng và giá trị riêng cho phép ta hiểu được bản chất của phép biến đổi A. Các vector riêng là các trục (axes) chính của dữ liệu, và các giá trị riêng cho biết mức độ quan trọng (độ lớn/phương sai) của mỗi trục đó.
Ví dụ cho bài toán phân tích phần tử chính (PCA) trong các mô hình NLP:

1. Tính ma trận hiệp phương sai Σ của dữ liệu.
2. Thực hiện phân rã $A\vec{v} = \lambda\vec{v}$ trên Σ.

3. Các vector riêng ($\vec{v}$) chính là các thành phần chính (principal components), là các trục mới, trực giao, đại diện cho các hướng có phương sai lớn nhất của dữ liệu.
4. Các giá trị riêng (λ) cho biết lượng phương sai (thông tin) mà mỗi vector riêng tương ứng nắm giữ.

Bằng cách chọn k vector riêng có k giá trị riêng lớn nhất, ta có thể giảm chiều dữ liệu (dimensionality reduction) từ n chiều xuống k chiều mà mất ít thông tin nhất. Ví dụ: Nén 1000 features của dữ liệu khách hàng xuống còn 10 features, hoặc nén 300 chiều Word2Vec xuống 2 chiều để vẽ đồ thị.

7 Phân rã giá trị suy biến (Singular Value Decomposition - SVD):

Phân rã giá trị suy biến là phép phân rã ma trận quan trọng, có thể áp dụng cho bất kỳ ma trận A nào (kể cả ma trận chữ nhật $m \times n$).

$$A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$$

- U: Ma trận trực giao chứa các vector suy biến trái (left-singular vectors).
- Σ: Ma trận đường chéo chứa các giá trị suy biến (singular values) σ_i.
- V^T: Ma trận chuyển vị của V (ma trận trực giao chứa các vector suy biến phải).

SVD có vai trò tìm ra các thành phần (components) hoặc chủ đề (topics) quan trọng nhất của dữ liệu. Các giá trị suy biến σ_i (giống như giá trị riêng) cho biết tầm quan trọng của mỗi thành phần.
Ví dụ:

- Trong thuật toán Latent Semantic Analysis - LSA: Cho A là ma trận TF-IDF [documents $\times$ terms]. SVD phân rã $A = U\Sigma V^T$. U biểu diễn mối liên hệ giữa [documents $\times$ topics]. V biểu diễn mối liên hệ giữa [terms $\times$ topics]. Σ cho biết tầm quan trọng của mỗi topic. Bằng cách giữ lại k giá trị suy biến lớn nhất (gọi là truncated SVD), ta nén được không gian ngữ nghĩa, giúp tìm các tài liệu tương đồng về chủ đề ngay cả khi chúng không dùng chung từ vựng.
- Trong Hệ gợi ý - Recommender Systems: Cho A là ma trận [users $\times$ items], với giá trị là rating. SVD tìm ra các yếu tố ẩn (latent factors) cho cả user và item (ví dụ: sở thích của user và thể loại của item). Phép $U\Sigma V^T$ có thể tái tạo lại ma trận A, bao gồm cả việc dự đoán (điền vào) các ô còn trống (các rating mà user chưa đánh giá).

8 Tensor: Tổng quát hóa của Vector và Ma trận

Tensor là một cấu trúc dữ liệu toán học có dạng đa chiều, là sự mở rộng tự nhiên của vector và ma trận:

- Vector: Tensor bậc 1 (1D).
- Ma trận: Tensor bậc 2 (2D).
- Tensor bậc n: mảng n chiều.

$$\mathscr{T} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n}$$

Tensor bậc 3 có dạng một khối lập phương dữ liệu

$$\mathscr{X} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$$

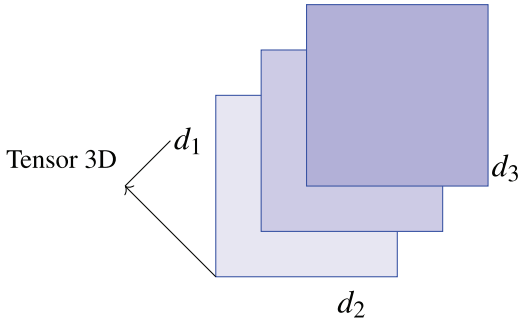
Ví dụ trong NLP: một câu gồm 10 từ, mỗi từ có vector embedding 300 chiều:

$$\mathscr{X} \in \mathbb{R}^{10 \times 300 \times 1}$$

Tensor bậc 4 rất phổ biến trong học sâu (Deep learning):

$$\mathscr{X} \in \mathbb{R}^{\text{batch} \times \text{channels} \times H \times W}$$

Tensor bậc n được biểu diễn bằng nhiều trục (axes), dùng cho dữ liệu đa phương thức hoặc phân tích nhiều chiều.



Hình 7: Minh họa Tensor bậc 3 như một khối dữ liệu 3 chiều.

Ví dụ Tensor trong NLP

Một batch gồm 32 câu, mỗi câu 20 từ, mỗi từ embedding 300 chiều:

$$\mathscr{X} \in \mathbb{R}^{32 \times 20 \times 300}$$

- 32: số câu trong batch.
- 20: số token trong mỗi câu.
- 300: kích thước embedding.

Tensor này là đầu vào của các mô hình RNN, Transformer, LSTM, Attention,...

8.1. Các phép toán cơ bản trên Tensor

Các phép toán dưới đây mở rộng tự nhiên từ vector và ma trận.

1. Cộng và trừ Tensor (Element-wise): Hai tensor cùng kích thước có thể cộng/trừ từng phần tử giống ma trận:

$$\mathscr{A} + \mathscr{B} = \{\mathscr{A}_{i_1,i_2,\ldots} + \mathscr{B}_{i_1,i_2,\ldots}\}$$

Một ứng dụng kinh điển của phép cộng tensor là trong các kết nối phần dư (residual connections) của các mạng như ResNet, trong đó Input X của một khối được cộng trực tiếp vào output $\mathscr{F}(X)$ của khối đó (giả định X và $\mathscr{F}(X)$ đã được xử lý để có cùng kích thước):

$$X_{\text{output}} = \mathscr{F}(X) + X_{\text{input}}$$

2. Nhân vô hướng (Tensor scaling)

$$s\mathscr{T} = \{s \cdot T_{i_1,i_2,\ldots,i_n}\}$$

Ví dụ ứng dụng: cập nhật trọng số mạng nơ-ron:

$$W_{\text{new}} = W_{\text{old}} - \eta \nabla W$$

Phép nhân vô hướng là một phép toán theo từng phần tử (element-wise), trong đó mọi phần tử của tensor được nhân với một giá trị vô hướng (scalar) duy nhất.

Cho một tensor $T \in \mathbb{R}^{d_1 \times \cdots \times d_n}$ và một số vô hướng $s \in \mathbb{R}$. Tensor kết quả $B = sT$ có các thành phần được định nghĩa là:

$$B_{i_1,i_2,\ldots,i_n} = s \cdot T_{i_1,i_2,\ldots,i_n}$$

Kết quả B là một tensor có kích thước (shape) giống hệt T.

Ví dụ ứng dụng trong học sâu: Cập nhật trọng số trong thuật toán Gradient Descent là một ứng dụng điển hình:

$$W_{\text{new}} = W_{\text{old}} - \eta \nabla W$$

Trong đó:

- W_{old} và ∇W (gradient) là các tensor (thường là ma trận hoặc tensor bậc cao hơn) có cùng kích thước.
- η (eta) là tốc độ học (learning rate), một giá trị vô hướng (ví dụ: 0.001).
- Phép toán $\eta \nabla W$ chính là tensor scaling: toàn bộ tensor gradient ∇W được *co* hoặc *dãn* (scale) bởi tốc độ học η trước khi nó được trừ khỏi tensor trọng số cũ.

8.2. Nhân Tensor–Ma trận (Mode-n Product)

Phép nhân tensor với ma trận (hay còn gọi là nhân mode-n) là một trong những phép toán nền tảng của đại số đa tuyến tính, tổng quát hóa phép nhân ma trận với vector. Phép toán này định nghĩa cách nhân một tensor T với một ma trận M dọc theo một trục (mode) n cụ thể của T.

Cho một tensor $T \in \mathbb{R}^{d_1 \times \cdots \times d_n \times \cdots \times d_N}$ và một ma trận $M \in \mathbb{R}^{J \times d_n}$.

Phép nhân mode-n của T với M, ký hiệu là $Y = T \times_n M$, là một tensor $Y \in \mathbb{R}^{d_1 \times \cdots \times J \times \cdots \times d_N}$. Kích thước của trục thứ n đã được thay đổi từ d_n thành J.

Các thành phần của Y được tính như sau:

$$(Y)_{i_1,\ldots,j,\ldots,i_N} = \sum_{k=1}^{d_n} T_{i_1,\ldots,k,\ldots,i_N} \cdot M_{j,k}$$

Ví dụ trong bài toán xử lý ảnh: Hãy tưởng tượng $T \in \mathbb{R}^{H \times W \times 3}$ là một ảnh màu (H: height, W: width, 3: kênh R,G,B). Chúng ta muốn biến đổi ảnh này thành ảnh xám (grayscale), tức là biến đổi trục 3 từ $d_3 = 3$ thành $J = 1$. Chúng ta cần một ma trận $M \in \mathbb{R}^{J \times d_n} = \mathbb{R}^{1 \times 3}$.
Ví dụ: $M = \begin{bmatrix} 0.299 & 0.587 & 0.114 \end{bmatrix}$.

Kết quả $Y = T \times_3 M$ sẽ là một tensor $\mathbb{R}^{H \times W \times 1}$ (ảnh xám). Tại mỗi pixel (i, j), phép toán là:

$$Y_{i,j,1} = \sum_{k=1}^{3} T_{i,j,k} \cdot M_{1,k} = T_{i,j,1} \cdot 0.299 + T_{i,j,2} \cdot 0.587 + T_{i,j,3} \cdot 0.114$$

Đây chính là phép nhân ma trận-vector tiêu chuẩn được áp dụng đồng thời trên $H \times W$ pixel.

8.3. Tích ngoài (Outer Product)

Tích ngoài là một phép toán cơ bản tạo ra một tensor có bậc (order) cao hơn từ hai (hoặc nhiều) tensor có bậc thấp hơn. Trường hợp phổ biến nhất là tích ngoài của hai vector, tạo ra một ma trận (tensor bậc 2).

Cho hai vector $a \in \mathbb{R}^{d_1}$ và $b \in \mathbb{R}^{d_2}$. Tích ngoài của chúng, ký hiệu là $T = a \otimes b$, là một ma trận $T \in \mathbb{R}^{d_1 \times d_2}$.

Các thành phần của T được định nghĩa là:

$$T_{ij} = a_i \cdot b_j$$

Trong đại số tuyến tính, phép toán này tương đương với phép nhân ma trận của một vector cột (a) với một vector hàng (b^T): $T = ab^T$.

Ví dụ: Cho $a = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ (kích thước 2×1) và $b = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$ (kích thước 3×1). Tích ngoài của chúng là:

$$T = a \otimes b = ab^T = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 3 & 4 & 5 \end{bmatrix} = \begin{bmatrix} 1 \cdot 3 & 1 \cdot 4 & 1 \cdot 5 \\ 2 \cdot 3 & 2 \cdot 4 & 2 \cdot 5 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 5 \\ 6 & 8 & 10 \end{bmatrix}$$

Kết quả là một ma trận kích thước 2×3.

8.4. Phép co Tensor (Tensor Contraction)

Phép co tensor là một phép toán cơ bản và tổng quát trong đại số đa tuyến tính, tổng quát hóa của nhiều phép toán quen thuộc (như tích vô hướng, nhân ma trận-vector, và nhân ma trận).

Phép co tensor là một phép toán thực hiện trên một hoặc hai tensor, trong đó ta thực hiện phép nhân và lấy tổng (sum-product) trên một hoặc nhiều cặp chỉ số (trục) được chỉ định.

Mỗi cặp chỉ số được co lại sẽ làm giảm tổng số bậc (order) của tensor.

Để định nghĩa phép co người ta thường sử dụng ký hiệu Einstein: bất kỳ chỉ số nào xuất hiện lặp lại trong một biểu thức đều được ngầm hiểu là phép lấy tổng trên chỉ số đó.

Một số phép toán cơ bản của phép co tensor:

- 1. Tích vô hướng (Dot Product): (Co 2 tensor bậc 1 $\rightarrow$ tensor bậc 0)
  - Công thức: $s = \sum_k a_k b_k$
  - Einsum: $s = a_k b_k$
- 2. Nhân Ma trận-Vector: (Co tensor bậc 2 và bậc 1 $\rightarrow$ tensor bậc 1)
  - Công thức: $y_i = \sum_k A_{ik} x_k$
  - Einsum: $y_i = A_{ik} x_k$
- 3. Vết (trace) của Ma trận: (Co trong một tensor bậc 2 $\rightarrow$ tensor bậc 0)

- Công thức: $\text{tr}(A) = \sum_k A_{kk}$
- Einsum: A_{kk}

Ví dụ ứng dụng: Attention trong Transformer dùng phép co tensor:

$$\text{Score} = QK^T$$

Đây là một ví dụ của phép co, được gọi là nhân ma trận theo batch (Batched Matrix Multiplication).

Trong đó:

- Tensor Query Q có kích thước (B, S_q, E)

  (Batch, Số token Query, Kích thước Embedding)

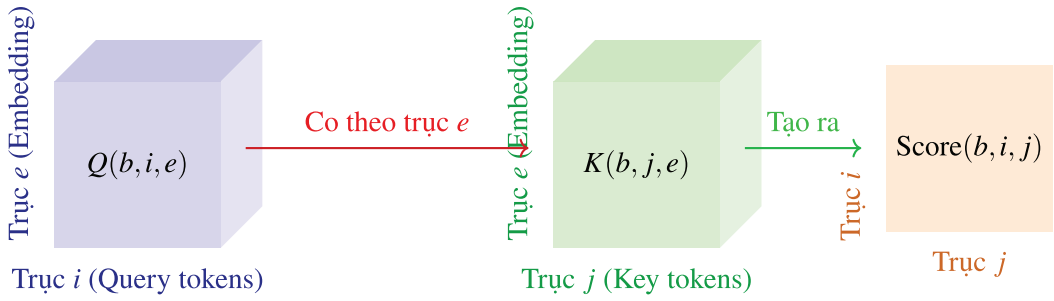
- Tensor Key K có kích thước (B, S_k, E)

  (Batch, Số token Key, Kích thước Embedding)

Để nhân, chúng ta cần chuyển vị K thành K^T có kích thước (B, E, S_k).

Phép toán $\text{Score} = QK^T$ tạo ra $\text{Score} \in (B, S_q, S_k)$, tương ứng biểu diễn dưới dạng thành phần để trực quan phép co:

$$\text{Score}_{b,i,j} = \sum_e Q_{b,i,e} \cdot K^T_{b,e,j}$$



Hình 8: Minh họa phép co tensor trong Attention: hai tensor $Q_{b,i,e}$ và $K_{b,j,e}$ được co theo trục embedding e, tạo ra ma trận điểm số $\text{Score}_{b,i,j} = \sum_e Q_{b,i,e} K_{b,j,e}$.

8.5. Hoán vị và Chuyển vị Tensor (Permute/Transpose)

Phép hoán vị (`permute`) là một phép toán cấu trúc (structural operation) tổng quát, cho phép sắp xếp lại thứ tự các trục (axes) của một tensor.

Cho một tensor X có N trục, được đánh chỉ số `(0, 1, ..., N-1)`. Phép `permute(p_0, p_1, ..., p_{N-1})` sẽ tạo ra một tensor X' mới, trong đó:

- Trục `0` của X' là trục p_0 của X.
- Trục `1` của X' là trục p_1 của X.
- ...
- Giá trị dữ liệu không thay đổi, chỉ có cách nhìn (view) dữ liệu thay đổi.

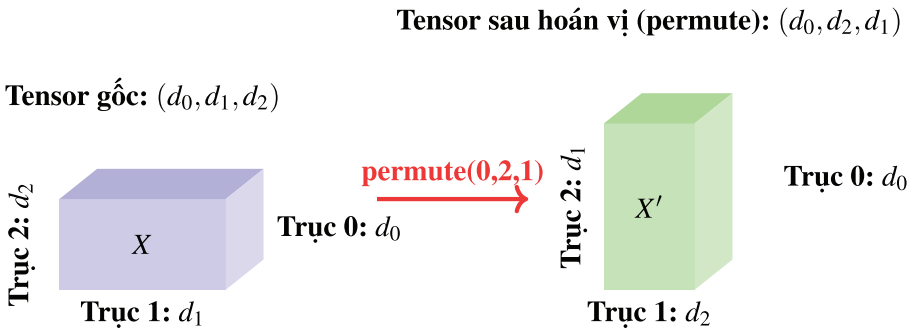
Ví dụ: Giả sử $X \in \mathbb{R}^{d_0 \times d_1 \times d_2}$ (tương ứng trục `0, 1, 2`). Khi thực hiện:

$$X' = X.\text{permute}(0,2,1)$$

Tensor mới X' sẽ có:

- Trục `0` mới là trục `0` cũ (kích thước d_0).
- Trục `1` mới là trục `2` cũ (kích thước d_2).
- Trục `2` mới là trục `1` cũ (kích thước d_1).

Do đó, $X' \in \mathbb{R}^{d_0 \times d_2 \times d_1}$.



Hình 9: Minh họa trực quan phép hoán vị trục `hoán vị(0,2,1)`. Các chiều có độ dài khác nhau để dễ thấy trục nào được chuyển sang vị trí nào.

Chuyển vị (transpose) là phép hoán vị đặc biệt, một trường hợp con của hoán vị, chỉ hoán đổi vị trí của hai trục được chỉ định (dimA và dimB) và giữ nguyên tất cả các trục khác.

8.6. Làm phẳng (Flatten)

Trong nhiều mô hình học sâu, đặc biệt là khi kết nối giữa các lớp tích chập (CNN) và các lớp tuyến tính (fully-connected), ta cần làm phẳng (flatten) một tensor nhiều chiều về dạng vector đặc trưng một chiều cho từng mẫu.

Cho một tensor đặc trưng bốn chiều

$$\mathscr{T} \in \mathbb{R}^{B \times C \times H \times W},$$

trong đó:

- B: kích thước batch (số mẫu trong một lần huấn luyện),
- C: số kênh đặc trưng (channels, ví dụ: số filter của CNN),
- H, W: chiều cao và chiều rộng của bản đồ đặc trưng (feature map).

Phép phẳng (flatten) sẽ giữ nguyên chiều batch và gom ba chiều còn lại thành một chiều duy nhất:

$$\mathscr{T} \in \mathbb{R}^{B\times C\times H\times W} \xrightarrow{\text{Flatten}} \mathscr{T}' \in \mathbb{R}^{B\times (C\cdot H\cdot W)}.$$

Về bản chất, đây là một phép tái sắp xếp hình dạng (reshape) thuần túy:

- Không thay đổi giá trị các phần tử,
- Không thay đổi thứ tự tuyến tính (trong bộ nhớ) của các phần tử,
- Chỉ thay đổi cách ta xem (view) dữ liệu: từ bản đồ đặc trưng 2D nhiều kênh sang một vector đặc trưng cho mỗi mẫu.

Ví dụ: Phép phẳng hoá Tensor trong Multi-Head Attention của mô hình Transformer.

Trong các mô hình Transformer nền tảng của BERT, GPT và nhiều kiến trúc hiện đại, cơ chế Multi-Head Attention (MHA) thực hiện phép tính chú ý song song qua nhiều đầu(heads). Sau khi tính toán, việc hợp nhất các đầu này thành một tensor có cấu trúc chuẩn cho các lớp tiếp theo đòi hỏi một thao tác phẳng hoá có kiểm soát, thực chất là thay đổi hình dạng (reshape/view) kết hợp với hoán vị trục (permute).

Với kích thước lô B, số đầu h, độ dài chuỗi S và kích thước mỗi đầu d_k, đầu ra tập hợp từ tất cả các head có dạng:

$$\mathscr{T}_{\text{heads}} \in \mathbb{R}^{B\times h\times S\times d_k}.$$

Tensor này giữ nguyên cấu trúc tuần tự (theo trục S) nhưng tách biệt không gian biểu diễn theo từng head.

Lớp Feed-Forward (FFN) trong transformer yêu cầu đầu vào có dạng ba chiều quen thuộc trong NLP:

$$\mathbb{R}^{B\times S\times E},$$

với

$$E = h \times d_k.$$

Do đó, hai trục h và d_k phải được kết hợp thành một chiều duy nhất, trong khi trục tuần tự S được bảo toàn tuyệt đối.

+ Quy trình phẳng hoá Tensor:

Quá trình phẳng hóa nhằm đưa tensor từ dạng bốn chiều về ba chiều gồm hai bước chính.

(a) Hoán vị trục (Permute)

Trước hết, sắp xếp lại các trục để đưa hai chiều cần ghép (head và chiều embedding của head) về nằm cạnh nhau:

$$\mathscr{T}_{\text{permuted}} = \mathscr{T}_{\text{heads}}.\text{permute}(0,2,1,3) \in \mathbb{R}^{B\times S\times h\times d_k}.$$

(b) Thay đổi hình dạng (Reshape)

Sau đó, ghép hai chiều cuối cùng thành một chiều biểu diễn hợp nhất:

$$\mathscr{T}_{\text{flat}} = \mathscr{T}_{\text{permuted}}.\text{reshape}(B, S, hd_k) \in \mathbb{R}^{B\times S\times E}.$$

Phẳng hoá trong trường hợp này được hiểu là phép gộp có kiểm soát, giữ nguyên ý nghĩa của trục chuỗi nhưng hợp nhất không gian đặc trưng của từng head.

Kết quả thu được:

$$\mathscr{T}_{\text{flat}} \xrightarrow{\text{Linear/FFN}} \mathbb{R}^{B\times S\times E},$$

là một biểu diễn vector giàu thông tin, tổng hợp đóng góp của tất cả các head. Điểm quan trọng là cấu trúc tuần tự vẫn được giữ nguyên, khác với phép phẳng hóa trong CNN (thường phẳng hoàn toàn về 1D). Đây thực chất là một phép thay đổi hình dạng (reshape) có mục đích rõ ràng đó là tổng hợp các hướng chú ý song song thành một không gian biểu diễn thống nhất.

Trong cơ chế Multi-Head Attention, phẳng hoá không phải là thao tác phá vỡ cấu trúc, mà là một quy trình chuẩn tắc gồm hoán vị và thay đổi hình dạng, nhằm hợp nhất các đặc trưng của nhiều head thành một biểu diễn chung. Phép biến đổi này vừa đảm bảo thông tin tuần tự được duy trì đầy đủ, vừa chuẩn hoá tensor cho tầng Linear/FFN phía sau, qua đó đóng vai trò nền tảng trong hoạt động của toàn bộ kiến trúc Transformer.

8.7. Tích chập (Convolution) trên Tensor

Tích chập một chiều (1D Convolution) là phép biến đổi tuyến tính cục bộ trên chuỗi dữ liệu, trong đó một bộ lọc (filter/kernel) được trượt qua từng vị trí của tensor đầu vào. Với đầu vào x và bộ lọc w có kích thước k, đầu ra tại vị trí i được xác định bởi:

$$y_i = \sum_{j=0}^{k-1} w_j x_{i+j}.$$

Phép toán này khai thác tính bất biến tịnh tiến (translation invariance), cho phép mô hình phát hiện các đặc trưng cục bộ xuất hiện tại bất kỳ vị trí nào trong chuỗi.

Trong thực tế, đầu vào thường là một tensor có nhiều kênh (channels). Với đầu vào $X \in \mathbb{R}^{L\times d}$ biểu diễn chuỗi gồm L phần tử, mỗi phần tử có kích thước d (ví dụ: chiều embedding), một bộ lọc 1D có dạng:

$$W \in \mathbb{R}^{k \times d}.$$

Khi đó đầu ra tại vị trí i là:

$$y_i = \sum_{p=0}^{k-1} \sum_{q=1}^{d} W_{p,q} X_{i+p,q}.$$

Nếu sử dụng m bộ lọc song song, ta thu được:

$$Y \in \mathbb{R}^{(L-k+1) \times m}.$$

Ứng dụng trong xử lý ngôn ngữ tự nhiên (NLP):

Một trong những mô hình tiêu biểu sử dụng tích chập trong NLP là TextCNN, trong đó:

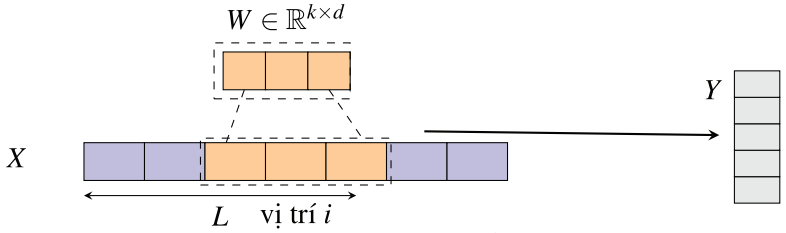
- Chuỗi từ được ánh xạ thành chuỗi embedding $X \in \mathbb{R}^{L \times d}$.
- Các bộ lọc có kích thước $k \in \{2,3,4\}$ (tương ứng với bi-gram, tri-gram, 4-gram).
- Mỗi bộ lọc trượt qua toàn bộ chuỗi embedding để phát hiện đặc trưng n-gram:

$$c_i = f(W * X) = f\left(\sum_{p,q} W_{p,q} X_{i+p,q}\right),$$

  trong đó f là hàm phi tuyến (ReLU).
- Sau đó áp dụng max-pooling trên toàn chuỗi để lấy đặc trưng nổi bật nhất.

Do đó, TextCNN thực hiện đúng tinh thần tích chập 1D: các bộ lọc k-gram được trượt dọc theo chuỗi embedding để rút trích đặc trưng cục bộ quan trọng.

1D Convolution trong TextCNN



chuỗi L phần tử, mỗi phần tử $\in \mathbb{R}^d$

$$y_i = \sum_{p=0}^{k-1} \sum_{q=1}^{d} W_{p,q} X_{i+p,q}, \quad Y \in \mathbb{R}^{(L-k+1) \times m}$$

TextCNN: lọc k-gram trượt trên chuỗi embedding để rút trích đặc trưng cục bộ.

Tensor là mô hình tổng quát và mạnh để biểu diễn dữ liệu nhiều chiều. Trong NLP, mọi mô hình hiện đại như Word2Vec, LSTM, Transformer, BERT đều hoạt động trên Tensor. Do đó, nắm vững Tensor chính là nền tảng để hiểu cách mô hình AI xử lý ngôn ngữ tự nhiên.

Kết luận

Vector, ma trận và tensor không chỉ là công cụ toán học, mà còn là ngôn ngữ của dữ liệu trong các mô hình NLP và AI. Việc hiểu sâu và vận dụng kiến thức toán Đại số tuyến tính sẽ giúp chúng ta xây dựng và tối ưu các mô hình như Word2Vec, BERT, hoặc Transformer một cách hiệu quả và có ý nghĩa. Nội dung của chương này đã giúp người học hiểu và hình dung cách mà các phép toán trên các cấu trúc dữ liệu đại số mô phỏng quá trình các mô hình AI/NLP lưu trữ, quan sát, học và hiểu được ngôn ngữ của con người.

Chương 2

CÁC BÀI TOÁN ĐẠI SỐ TUYẾN TÍNH
TRONG TRÍ TUỆ NHÂN TẠO VÀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Giới thiệu 39
1 Bài toán về vector hóa dữ liệu ngôn ngữ tự nhiên 39
1.1. Bài tập 1. Biểu diễn One-hot encoding và ma trận câu) . . . 39
1.2. Bài tập 2. Bag-of-Words cho tập văn bản nhỏ 40
1.3. Bài tập 3. TF–IDF và tầm quan trọng từ 42
1.4. Bài tập 4. Embedding lookup và ma trận câu 43
1.5. Bài tập 5. So sánh BoW và embedding qua độ tương đồng . 44
2 Các phép toán trên Vector và Ma trận 46
2.1. Bài tập 6. Tích vô hướng và Cosine cơ bản 46
2.2. Bài tập 7. Nhân vô hướng và co giãn (scaling) 48
2.3. Bài tập 8. Chuẩn hóa và cosine 50
2.4. Bài tập 9. Nhân ma trận–vector và biến đổi tuyến tính 51
2.5. Bài tập 10. Nhân hai ma trận 53
3 Phương sai, hiệp phương sai, PCA 54
3.1. Bài tập 11. Phương sai trong AI và NLP 54
3.2. Bài tập 12. Hiệp phương sai giữa hai từ (Covariance) 56
3.3. Bài tập 13. Ma trận hiệp phương sai (Covariance Matrix) . . 59
4 Giá trị riêng (Eigenvalue) và Vector riêng (Eigenvector), SVD, LSA 62
4.1. Bài tập 14. Giá trị riêng và vector riêng 62
4.2. Bài tập 15. Phân rã giá trị suy biến(SVD – Singular Value Decomposition) 65
4.3. Bài tập 16. PCA/LSA thu nhỏ (mini) với SVD rút ngọn (`TruncatedSVD`) 68
5 Các phép toán với Tensor, Convolution, Attention 72
5.1. Bài tập 17. Hoán vị trục (Permute) trong Multi-Head Attention 72
5.2. Bài tập 18. Định hình lại (Reshape/View) để gộp head trong Multi-Head Attention 74
5.3. Bài tập 19. Tích chập 1D trong TextCNN cho NLP 75
5.4. Bài tập 20. TextCNN thu nhỏ: tích chập trên embedding và max-pooling theo thời gian 78
Kết luận 81

Giới thiệu

Ở chương trước, chúng ta đã được hệ thống hóa kiến thức về Đại số tuyến tính bao gồm các cấu trúc đại số như Vector, Ma trận, Tensor cùng các phép toán và diễn giải tính ứng dụng của các cấu trúc đại số này trong AI và NLP. Tiếp theo là hệ thống bài toán nhằm giúp cho người học củng cố các kiến thức đại số tuyến tính đã được tiếp cận ở phần lý thuyết, đặc biệt các nội dung sau:

- Biểu diễn dữ liệu ngôn ngữ tự nhiên (vector hóa văn bản).
- Các phép toán vector, ma trận, tensor cốt lõi.
- Ứng dụng trong các mô hình NLP: TF-IDF, Word Embeddings, TextCNN, Attention, Transformer.

Mỗi bài toán được trình bày theo cấu trúc gồm ba phần: đề bài, lời giải chi tiết (bao gồm thuật toán/phân tích cho các bài ứng dụng), nhận xét và liên hệ NLP, hình vẽ trực quan và mã Python minh họa.

1 Bài toán về vector hóa dữ liệu ngôn ngữ tự nhiên

1.1. Bài tập 1. Biểu diễn One-hot encoding và ma trận câu)

Đề bài. Xét từ vựng (vocabulary) nhỏ:

$$V = \{\text{“tôi”}, \text{“yêu”}, \text{“VKU”}, \text{“Đà Nẵng”}\}.$$

(a) Xây dựng biểu diễn *one-hot* cho mỗi từ.
(b) Biểu diễn câu *“tôi yêu VKU”* dưới dạng ma trận one-hot kích thước $L \times |V|$.
(c) Vẽ hình minh họa trực quan ma trận này.
(d) Viết mã Python tạo ma trận one-hot cho câu trên.

Lời giải.
One-hot là vector biểu diễn dữ liệu rời rạc và rất thưa (sparse) chỉ có một phần tử có giá trị 1, tất cả phần tử còn lại có giá trị 0. Nhược điểm lớn nhất của One-hot là không mang thông tin ngữ nghĩa. Tuy nhiên, One-hot có vai trò làm nền tảng cho lớp Embedding trong các mạng nơ-ron (neural): ma trận X (chứa chỉ số) được dùng để tra cứu (lookup) các vector dày đặc (dense) trong một ma trận trọng số $W_E \in \mathbb{R}^{|V| \times d_{\text{model}}}$.

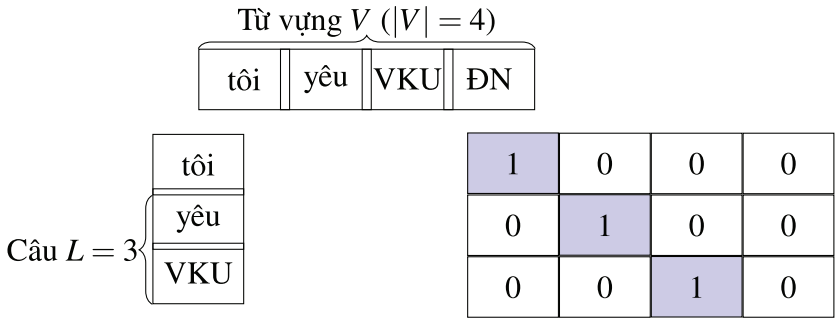
(a) Đánh chỉ số: “tôi” $\to 0$, “yêu” $\to 1$, “VKU” $\to 2$, “Đà Nẵng” $\to 3$. $|V| = 4$.
Biểu diễn one-hot:

$$\text{“tôi”} = [1,0,0,0],\ \text{“yêu”} = [0,1,0,0],\ \text{“VKU”} = [0,0,1,0],\ \text{“Đà Nẵng”} = [0,0,0,1].$$

(b) Câu "tôi yêu VKU" ($L = 3$):

$$X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 4}.$$

(c) Minh hoạ:



Hình 1: Ma trận one-hot $X \in \mathbb{R}^{3\times 4}$ cho câu "tôi yêu VKU"

(d) Mã Python:

```
vocab = ["toi", "yeu", "VKU", "Da Nang"]
word2idx = {w: i for i, w in enumerate(vocab)}  # map tu
    -> chi so

sentence = ["toi", "yeu", "VKU"]
L = len(sentence)
V = len(vocab)

import numpy as np
X = np.zeros((L, V), dtype=int)

for i, w in enumerate(sentence):
    if w in word2idx:
        j = word2idx[w]
        X[i, j] = 1

print("Tu vung:", word2idx)
print("Ma tran one-hot (L x V):\n", X)
```

Mã lệnh Python 1: One-hot encoding cho câu "toi yeu VKU"

1.2. Bài tập 2. Bag-of-Words cho tập văn bản nhỏ

Đề bài. Xét 3 câu tiếng Việt đã được tách từ: d_1: "tôi yêu VKU", d_2: "tôi học tại VKU", d_3: "Đà Nẵng có trường VKU".

(a) Xây dựng từ vựng V (sắp xếp alpha B).
(b) Xây dựng ma trận Bag-of-Words (BoW) $M \in \mathbb{R}^{3\times|V|}$ (dựa trên đếm tần suất).
(c) Minh hoạ ma trận M dưới dạng bảng (sử dụng `booktabs`).

Lời giải.

BoW ánh xạ một câu độ dài L thành một vector cố định trong $\mathbb{R}^{|V|}$, nhưng không mã hoá thứ tự từ, thường tạo biểu diễn thưa (sparse), và xem mọi từ có cùng mức đóng góp nên chưa phản ánh được mức độ quan trọng hay ngữ nghĩa của từng từ.

(a) Từ vựng $V = \{\text{có, Đà Nẵng, học, tại, tôi, trường, VKU, yêu}\}$.
$(|V| = 8)$.

(b) Đếm tần suất: d_1: tôi(1), yêu(1), VKU(1). d_2: tôi(1), học(1), tại(1), VKU(1). d_3: Đà Nẵng(1), có(1), trường(1), VKU(1).

$$M = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 8}.$$

(c) Bảng BoW:

Bảng 1: Ma trận Bag-of-Words (BoW) cho 3 văn bản

	Từ vựng (Vocabulary V)							
	có	Đà Nẵng	học	tại	tôi	trường	VKU	yêu
d_1	0	0	0	0	1	0	1	1
d_2	0	0	1	1	1	0	1	0
d_3	1	1	0	0	0	1	1	0

(d) Mã Python (dùng `CountVectorizer`):

```
from sklearn.feature_extraction.text import
   CountVectorizer
docs = [
    "toi yeu VKU",              # d1
    "toi hoc tai VKU",          # d2
    "Da Nang co truong VKU"  # d3
]
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(docs)
print("Tu vung (feature names):", vectorizer.
   get_feature_names_out())
print("Ma tran BoW (N_docs x |V|):\n", X.toarray())
```

Mã lệnh Python 2: Tính Bag-of-Words bằng scikit-learn

1.3. Bài tập 3. TF–IDF và tầm quan trọng từ

Đề bài. Với cùng 3 văn bản d_1, d_2, d_3 và từ vựng V ở Bài 2.
(a) Tính DF và IDF cho hai từ: $t_1 =$ "VKU" và $t_2 =$ "tôi". Sử dụng công thức (đã làm trơn):

$$\text{IDF}(t) = \log \frac{N+1}{\text{DF}(t)+1} + 1, \quad (\text{với } N = 3).$$

(b) Tính giá trị TF–IDF (TF $\times$ IDF) cho từ "tôi" trong d_1 và từ "VKU" trong d_1. (Sử dụng TF thô).
(c) Vẽ hình minh họa quan hệ giữa DF và IDF.

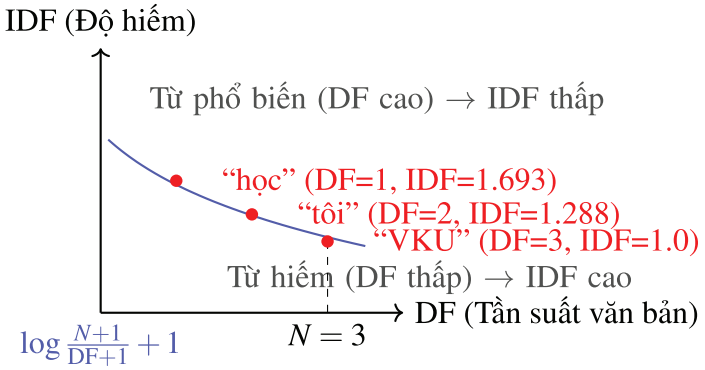
Lời giải.

TF–IDF là một bước cải tiến quan trọng so với BoW vì nó giảm trọng số các từ xuất hiện trong nhiều văn bản (DF cao) và tăng trọng số các từ hiếm (DF thấp), từ đó làm nổi bật tính phân biệt giữa các văn bản.

(a) $N = 3$. DF(VKU) $= 3$ (xuất hiện trong d_1, d_2, d_3).
DF(tôi) $= 2$ (xuất hiện trong d_1, d_2).
IDF(VKU) $= \log \frac{3+1}{3+1} + 1 = \log(1) + 1 = 1.0$.
IDF(tôi) $= \log \frac{3+1}{2+1} + 1 = \log \frac{4}{3} + 1 \approx 1.288$.

(b) TF("tôi", d_1) $= 1$.
TF("VKU", d_1) $= 1$.
TF-IDF("tôi", d_1) $= 1 \times 1.288 = 1.288$.
TF-IDF("VKU", d_1) $= 1 \times 1.0 = 1.0$.

(c) Minh họa:



Hình 2: Quan hệ tỉ lệ nghịch giữa Document Frequency (DF) và IDF

1.4. Bài tập 4. Embedding lookup và ma trận câu

Đề bài. Giả sử ta có bảng tra cứu (lookup table) embedding 3 chiều ($d_{\text{model}} = 3$):

$$\text{Embed}(\text{“tôi”}) \to [0.1, 0.8, 0.1], \quad \text{Embed}(\text{“yêu”}) \to [0.9, 0.1, 0.2]$$
$$\text{Embed}(\text{“VKU”}) \to [0.1, 0.1, 0.9], \quad \text{Embed}(\text{“Đà Nẵng”}) \to [0.2, 0.2, 0.8]$$

(a) Biểu diễn câu “tôi yêu VKU” (dài $L = 3$) thành ma trận embedding $X_{\text{emb}} \in \mathbb{R}^{3\times 3}$.
(b) Vẽ sơ đồ trực quan (3D) minh họa các vector embedding này.
(c) Viết mã Python xây dựng ma trận embedding (giả lập lookup).

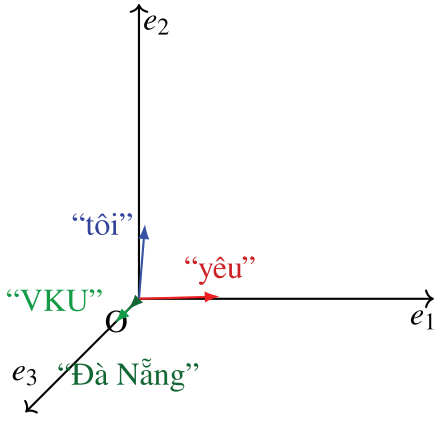
Lời giải.

Embedding là biểu diễn véc-tơ dày đặc (dense), được học từ dữ liệu và mã hoá thông tin ngữ nghĩa. Trong hình dưới, các véc-tơ của “VKU” và “Đà Nẵng” nằm gần nhau, phản ánh mức độ liên quan theo ngữ cảnh, đặc điểm mà biểu diễn one-hot không có do các véc-tơ luôn trực giao. Ma trận embedding $X_{\text{emb}} \in \mathbb{R}^{L\times d}$ là dạng đầu vào tiêu chuẩn cho các mô hình như RNN, CNN và Transformer.

(a) Xếp chồng các vector:

$$X_{\text{emb}} = \begin{bmatrix} \text{Embed}(\text{“tôi”}) \\ \text{Embed}(\text{“yêu”}) \\ \text{Embed}(\text{“VKU”}) \end{bmatrix} = \begin{bmatrix} 0.1 & 0.8 & 0.1 \\ 0.9 & 0.1 & 0.2 \\ 0.1 & 0.1 & 0.9 \end{bmatrix} \in \mathbb{R}^{3\times 3}.$$

(b) Minh họa 3D:



Hình 3: Các vector embedding trong không gian 3D.

(c) Mã Python: (Đã bỏ dấu)

```
import numpy as np

# Bang tra cuu (ma tran |V| x d)
embeddings_table = {
```

```
    "toi": np.array([0.1, 0.8, 0.1]),
    "yeu": np.array([0.9, 0.1, 0.2]),
    "VKU": np.array([0.1, 0.1, 0.9]),
    "Da Nang": np.array([0.2, 0.2, 0.8])
}
# Cau can bieu dien
sentence = ["toi", "yeu", "VKU"]

# Xay dung ma tran L x d
X_emb = np.array([embeddings_table[w] for w in sentence])

print("Ma tran embedding (L x d_model):\n", X_emb)
print("Kich thuoc:", X_emb.shape) # (3, 3)
```

Mã lệnh Python 3: Embedding lookup (giả lập bằng dict và numpy)

1.5. Bài tập 5. So sánh BoW và embedding qua độ tương đồng

Đề bài. Giả sử ta có:

- $d_1 =$ "tôi yêu VKU", $d_2 =$ "tôi thích VKU".
- BoW trên $V = \{\text{tôi}, \text{yêu}, \text{thích}, \text{VKU}\}$.
- Embedding 2D:
  Embed("yêu") $= [1,1]$, Embed("thích") $= [0.9,1]$,
  Embed("tôi") $= [0,0.1]$, Embed("VKU") $= [0.5,0.5]$.

(a) Tính cosine similarity giữa $\vec{d}_1$ và $\vec{d}_2$ (BoW).
(b) Tính vector trung bình (mean pooling) $\vec{e}(d_1)$ và $\vec{e}(d_2)$ (Embedding).
(c) Tính cosine similarity giữa $\vec{e}(d_1)$ và $\vec{e}(d_2)$.
(d) Vẽ 2 hình minh họa và viết code Python.

Lời giải.

(a) Không gian BoW ($|V| = 4$):
$\vec{d}_1 = [1,1,0,1]$, $\vec{d}_2 = [1,0,1,1]$. $\vec{d}_1 \cdot \vec{d}_2 = 1+0+0+1 = 2$. $\|\vec{d}_1\| = \sqrt{3}$, $\|\vec{d}_2\| = \sqrt{3}$.

$\cos(\vec{d}_1, \vec{d}_2) = \frac{2}{\sqrt{3}\cdot\sqrt{3}} = \frac{2}{3} \approx 0.667$.

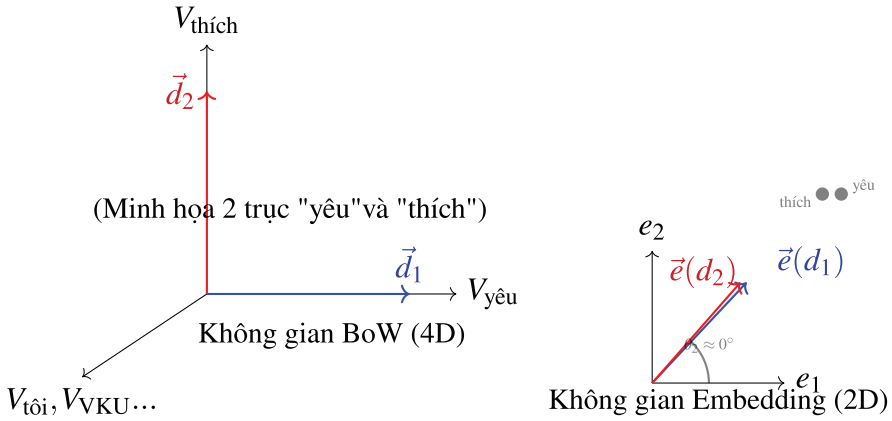
(b) Vector trung bình embedding (không gian 2D):
$\vec{e}(d_1) = \frac{1}{3}([0,0.1]+[1,1]+[0.5,0.5]) = \frac{1}{3}[1.5,1.6] \approx [0.5,0.533]$.
$\vec{e}(d_2) = \frac{1}{3}([0,0.1]+[0.9,1]+[0.5,0.5]) = \frac{1}{3}[1.4,1.6] \approx [0.467,0.533]$.

(c) Cosine Similarity (Embedding):
$\vec{e}(d_1) = [1.5,1.6]/3$, $\vec{e}(d_2) = [1.4,1.6]/3$.
$\vec{e}(d_1) \cdot \vec{e}(d_2) = \frac{1}{9}(1.5\cdot 1.4 + 1.6\cdot 1.6) = \frac{4.66}{9}$.
$\|\vec{e}(d_1)\| = \frac{1}{3}\sqrt{1.5^2+1.6^2} = \frac{\sqrt{4.81}}{3}$.

$\|\vec{e}(d_2)\| = \frac{1}{3}\sqrt{1.4^2 + 1.6^2} = \frac{\sqrt{4.52}}{3}$.
$\cos(\vec{e}(d_1), \vec{e}(d_2)) = \frac{4.66/9}{(\sqrt{4.81}/3)\cdot(\sqrt{4.52}/3)} = \frac{4.66}{\sqrt{4.81}\cdot\sqrt{4.52}} \approx 0.999.$

Kết quả cos ≈ 0.667 (BoW) và cos ≈ 0.999 (Embedding) cho thấy sự khác biệt cốt lõi. BoW xem "yêu" và "thích" là 2 đặc trưng (features) hoàn toàn khác nhau. Embedding (được huấn luyện tốt) sẽ đặt các từ đồng nghĩa gần nhau, phản ánh đúng ngữ nghĩa hai câu ví dụ trên rất tương đồng.

(d) Minh họa trực quan:



Hình 4: So sánh độ tương đồng.

Hình trái là không gian BoW xem "yêu" và "thích" là trực giao ($\theta_1 = 90°$ trong 2 trục này). Hình phải là không gian Embedding đặt "yêu" và "thích" gần nhau, do đó vector trung bình của 2 câu gần như trùng nhau ($\theta_2 \approx 0$).

(e) Mã Python:

```
import numpy as np
from numpy.linalg import norm

# (a) Khong gian BoW
d1_bow = np.array([1, 1, 0, 1])
d2_bow = np.array([1, 0, 1, 1])
cos_bow = np.dot(d1_bow, d2_bow) / (norm(d1_bow) * norm(
   d2_bow))
print(f"BoW Cosine Similarity: {cos_bow:.3f}") # 0.667

# (b, c) Khong gian Embedding
emb = {
    "toi": np.array([0, 0.1]), "VKU": np.array([0.5,
   0.5]),
```

```
    "yeu": np.array([1, 1]), "thich": np.array([0.9, 1])
}
e_d1 = (emb["toi"] + emb["yeu"] + emb["VKU"]) / 3
e_d2 = (emb["toi"] + emb["thich"] + emb["VKU"]) / 3

cos_emb = np.dot(e_d1, e_d2) / (norm(e_d1) * norm(e_d2))
print(f"Embedding Cosine Similarity: {cos_emb:.3f}") #
   0.999
```

Mã lệnh Python 4: Tinh cosine similarity (Bai 5)

2 Các phép toán trên Vector và Ma trận

2.1. Bài tập 6. Tích vô hướng và Cosine cơ bản

Đề bài. Giả sử $\vec{a}$ và $\vec{b}$ là hai vector embedding 3 chiều biểu diễn cho hai câu tiếng Việt khác nhau, được trích xuất từ một mô hình NLP đơn giản (ví dụ: mô hình sentence embedding thu nhỏ).
Cho $\vec{a} = [2, 3, -1]$ và $\vec{b} = [1, -1, 4]$.
(a) Tính tích vô hướng $\vec{a} \cdot \vec{b}$.
(b) Tính chuẩn L_2 $\|\vec{a}\|$ và $\|\vec{b}\|$.
(c) Tính cosine similarity $\cos(\theta)$ giữa $\vec{a}$ và $\vec{b}$.
(d) Vẽ hình minh họa 3D và viết code Python (minh hoạ cách so sánh độ giống nhau giữa hai câu dựa trên embedding).

Lời giải.

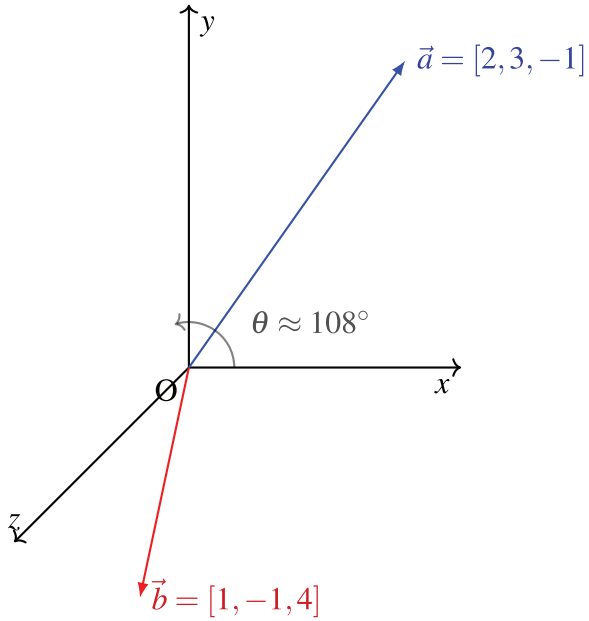
(a) $\vec{a} \cdot \vec{b} = (2 \cdot 1) + (3 \cdot (-1)) + ((-1) \cdot 4) = 2 - 3 - 4 = -5.$

(b) $\|\vec{a}\| = \sqrt{2^2 + 3^2 + (-1)^2} = \sqrt{14}$. $\|\vec{b}\| = \sqrt{1^2 + (-1)^2 + 4^2} = \sqrt{18}.$

(c) $\cos(\theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|} = \frac{-5}{\sqrt{14} \cdot \sqrt{18}} = \frac{-5}{\sqrt{252}} \approx -0.315.$

Tích vô hướng âm (-5) và $\cos(\theta) < 0$ cho thấy $\vec{a}$ và $\vec{b}$ có hướng ngược nhau. Trong NLP, nếu embedding của hai từ (ví dụ: "nóng" và "lạnh") có cosine âm, điều đó cho thấy chúng là trái nghĩa (antonyms).

(d) Minh họa 3D:



Hình 5: Minh họa $\vec{a}$ và $\vec{b}$ trong không gian 3D. Kết quả $\cos(\theta) < 0$ cho thấy chúng tạo một góc tù (hướng ngược nhau).

(e) Mã Python:

```
import numpy as np
from numpy.linalg import norm

a = np.array([2, 3, -1])
b = np.array([1, -1, 4])

# (a) Tich vo huong
dot_product = np.dot(a, b)
print(f"Tich vo huong: {dot_product}") # -5

# (b) Chuan L2
norm_a = norm(a)
norm_b = norm(b)
print(f"Norm a: {norm_a:.3f}") # sqrt(14) = 3.742
print(f"Norm b: {norm_b:.3f}") # sqrt(18) = 4.243

# (c) Cosine Similarity
cos_sim = dot_product / (norm_a * norm_b)
print(f"Cosine Similarity: {cos_sim:.3f}") # -0.315
```

Mã lệnh Python 5: Tích vô hướng và Cosine (Bài 6)

2.2. Bài tập 7. Nhân vô hướng và co giãn (scaling)

Đề bài. Giả sử $\vec{v}$ là vector embedding 3 chiều của một câu trong không gian biểu diễn của một mô hình NLP (ví dụ: sentence embedding). Hệ số vô hướng $s = 0.5$ được dùng để làm *giảm cường độ* embedding này (ví dụ khi chuẩn hoá hoặc điều chỉnh trọng số đặc trưng).
Cho vector embedding $\vec{v} = [1, -2, 3]$ và vô hướng (scalar) $s = 0.5$.
(a) Tính $s\vec{v}$.
(b) So sánh chuẩn của $\vec{v}$ và $s\vec{v}$ (giải thích ý nghĩa việc thu nhỏ độ dài embedding).
(c) Vẽ hình minh họa và viết code Python.

Lời giải. (a) Ta có

$$\vec{v} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}, \qquad s = 0.5.$$

Khi đó

$$s\vec{v} = 0.5 \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ -1 \\ 1.5 \end{bmatrix}.$$

(b) Chuẩn L_2 của $\vec{v}$ là

$$\|\vec{v}\| = \sqrt{1^2 + (-2)^2 + 3^2} = \sqrt{1+4+9} = \sqrt{14}.$$

Chuẩn L_2 của $s\vec{v}$:

$$\|s\vec{v}\| = \left\| \begin{bmatrix} 0.5 \\ -1 \\ 1.5 \end{bmatrix} \right\| = \sqrt{0.5^2 + (-1)^2 + 1.5^2} = \sqrt{0.25 + 1 + 2.25} = \sqrt{3.5}$$

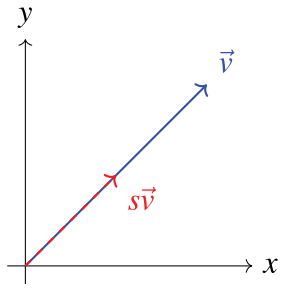
$$= 0.5\sqrt{14} = |s| \, \|\vec{v}\|.$$

Như vậy, nhân embedding với vô hướng $s = 0.5$ làm giảm một nửa độ dài (magnitude) của vector embedding, nhưng không thay đổi hướng. Trong bài toán NLP, điều này tương ứng với việc *giảm cường độ* biểu diễn của câu, ví dụ khi ta chuẩn hoá hoặc điều chỉnh trọng số đặc trưng (feature weighting).

Phép nhân vô hướng (scaling) chỉ thay đổi độ dài (magnitude) của vector. Trong Gradient Descent, learning rate η (vô hướng) co dãn vector gradient ∇J để quyết định bước cập nhật trọng số: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla J$.

(c) Minh hoạ bằng cách chiếu embedding 3 chiều xuống mặt phẳng 2D (giữ nguyên ý tưởng *vector gốc* và "vector sau khi co dãn):

Mã Python minh hoạ tính toán và vẽ hai vector embedding trong 3D:



Hình 6: Minh họa ý tưởng scaling embedding (chiếu từ 3D xuống 2D).

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  #  c n  cho  v
    3D

# (a)
v = np.array([1, -2, 3], dtype=float)
s = 0.5
sv = s * v

print("v       =", v)
print("s * v   =", sv)

# (b)
norm_v = np.linalg.norm(v)
norm_sv = np.linalg.norm(sv)

print("||v||       =", norm_v)
print("||s * v||   =", norm_sv)
print(" T    l   ||s*v|| / ||v|| =", norm_sv / norm_v)

# (c)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')


ax.quiver(0, 0, 0,
          v[0], v[1], v[2],
          length=1, normalize=False)


ax.quiver(0, 0, 0,
          sv[0], sv[1], sv[2],
          length=1, normalize=False,
          linestyle="dashed")
```

```
ax.set_xlim(0, max(v[0], sv[0]) + 1)
ax.set_ylim(min(v[1], sv[1]) - 1, 1)
ax.set_zlim(0, v[2] + 1)

ax.set_xlabel("Dim 1")
ax.set_ylabel("Dim 2")
ax.set_zlabel("Dim 3")
ax.set_title("Scaling embedding: v và s * v")

plt.show()
```

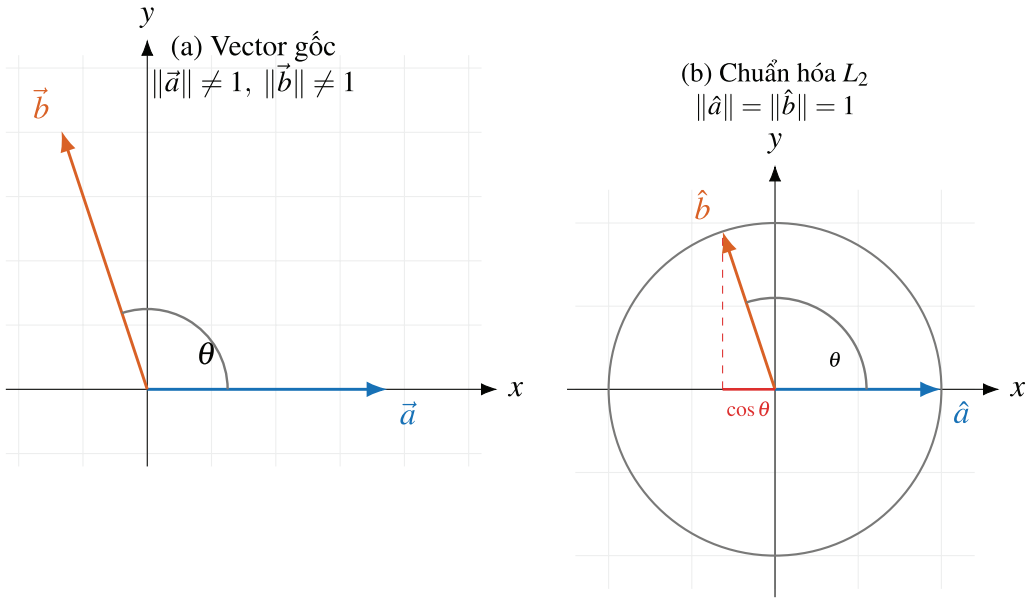
2.3. Bài tập 8. Chuẩn hóa và cosine

Đề bài. Trong các mô hình NLP, độ giống nhau giữa hai từ/câu thường được đo bằng cosine similarity giữa các vector embedding $\vec{a}$ và $\vec{b}$.
Chứng minh rằng $\cos(\theta) = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$ chính là tích vô hướng (dot product) giữa hai vector embedding đã được chuẩn hóa L_2 (unit vectors): $\hat{a} = \frac{\vec{a}}{\|\vec{a}\|}$ và $\hat{b} = \frac{\vec{b}}{\|\vec{b}\|}$.
Viết code Python minh họa với hai vector embedding ví dụ.

Lời giải. Xét tích vô hướng của $\hat{a}$ và $\hat{b}$:

$$\hat{a}\cdot\hat{b} = \left(\frac{\vec{a}}{\|\vec{a}\|}\right)\cdot\left(\frac{\vec{b}}{\|\vec{b}\|}\right) = \frac{1}{\|\vec{a}\|}\frac{1}{\|\vec{b}\|}(\vec{a}\cdot\vec{b}) = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \cos(\theta).$$

Hai kết quả là giống hệt nhau. Khi so sánh embedding, nếu chúng ta chuẩn hóa (normalize) tất cả các embedding về vector đơn vị (độ dài 1), thì việc tính cosine similarity tốn kém (phép chia) nên chúng ta sẽ quy về việc tính tích vô hướng (chỉ nhân và cộng) nhanh hơn nhiều.



Hình 7: So sánh trước/sau chuẩn hóa L_2: sau chuẩn hóa, $\hat{a} \cdot \hat{b} = \cos\theta$ và phần chiếu của $\hat{b}$ lên trục $\hat{a}$ có độ dài đúng bằng $\cos\theta$ (có thể âm).

```python
import numpy as np
from numpy.linalg import norm

a = np.array([2, 3, -1])
b = np.array([1, -1, 4])

# Cach 1: Tinh Cosine (nhu Bai 6)
cos_sim = np.dot(a, b) / (norm(a) * norm(b))
print(f"Cosine Similarity (cach 1): {cos_sim:.3f}")

# Cach 2: Chuan hoa roi tinh Tich vo huong
a_hat = a / norm(a)
b_hat = b / norm(b)
# print(f"Norm a_hat: {norm(a_hat)}") # Se bang 1.0

dot_normalized = np.dot(a_hat, b_hat)
print(f"Dot Product cua vector chuan hoa (cach 2): {
   dot_normalized:.3f}")
```

Mã lệnh Python 6: Cosine vs. Dot product của vector chuẩn hóa (Bài 8)

2.4. Bài tập 9. Nhân ma trận–vector và biến đổi tuyến tính

Mã Python (sử dụng dữ liệu Bài 6).

Đề bài. Xét một tầng tuyến tính (fully-connected layer) rất đơn giản trong một mô hình xử lý ngôn ngữ tự nhiên (NLP). Giả sử mỗi câu được biểu diễn bởi một vector đặc trưng 2 chiều

$$\vec{x} = \begin{bmatrix} 1 \\ 2 \end{bmatrix},$$

trong đó mỗi thành phần có thể xem như một đặc trưng trích xuất từ embedding (ví dụ: mức độ liên quan đến chủ đề "AI" và "data"). Tầng tuyến tính này có ma trận trọng số

$$A = \begin{bmatrix} 2 & -1 \\ 1 & 3 \end{bmatrix},$$

dùng để biến đổi vector đặc trưng đầu vào $\vec{x}$ thành một biểu diễn ẩn mới $\vec{y}$ (hidden representation), phục vụ cho các tác vụ như phân loại cảm xúc, phân loại chủ đề, v.v.

(a) Tính vector đầu ra $\vec{y} = A\vec{x}$.

(b) Vẽ hình minh họa phép biến đổi tuyến tính này trong mặt phẳng đặc trưng (cho thấy cách vị trí của điểm biểu diễn câu thay đổi trước và sau tầng tuyến tính).

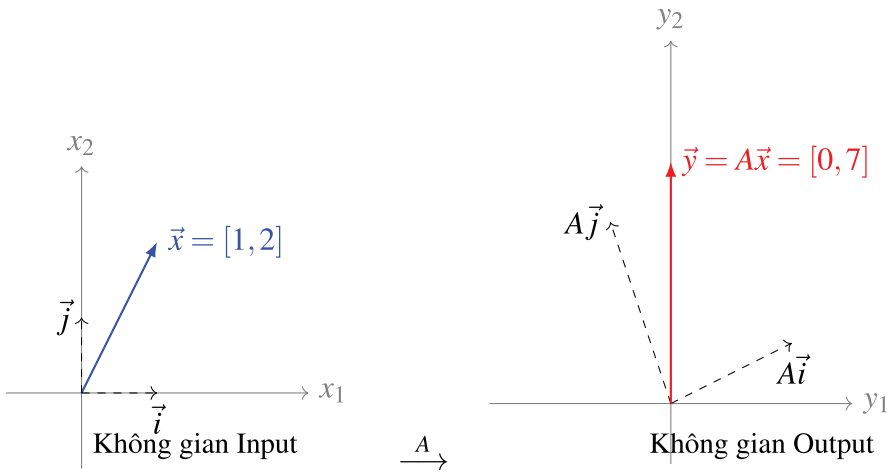
(c) Viết code Python mô phỏng tầng tuyến tính nói trên (tính $\vec{y}$ từ A và $\vec{x}$).

Lời giải.

(a)

$$\vec{y} = A\vec{x} = \begin{bmatrix} 2 & -1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} (2\cdot 1) + ((-1)\cdot 2) \\ (1\cdot 1) + (3\cdot 2) \end{bmatrix} = \begin{bmatrix} 0 \\ 7 \end{bmatrix}.$$

(b) Minh họa: Phép biến đổi A biến không gian: $\vec{i} = [1,0]^T \to [2,1]^T$ (cột 1) và $\vec{j} = [0,1]^T \to [-1,3]^T$ (cột 2). $A\vec{x} = 1(A\vec{i}) + 2(A\vec{j}) = 1[2,1]^T + 2[-1,3]^T = [0,7]^T$.



Hình 8: Phép nhân ma trận $A\vec{x}$ là một phép biến đổi tuyến tính, bẻ cong không gian và ánh xạ $\vec{x}$ thành $\vec{y}$.

Toàn bộ mạng nơ-ron (trừ các hàm kích hoạt) về cơ bản là một chuỗi các phép biến đổi tuyến tính (nhân ma trận trọng số A) và phép tịnh tiến (cộng vector bias $\vec{b}$), trong đó $A\vec{x}$ là phép toán cốt lõi của AI.

(c) Mã Python:

```python
import numpy as np
A = np.array([
    [2, -1],
    [1,  3]
])
x = np.array([1, 2])

# Dung np.dot() hoac toan tu @
y = np.dot(A, x)
# y = A @ x # Tuong duong

print(f"y = A*x = {y}") # [0, 7]
```

Mã lệnh Python 7: Nhân Ma trận và Vector (Bài 9)

2.5. Bài tập 10. Nhân hai ma trận

Đề bài. Trong cơ chế Self-Attention của Transformer, ta thường phải nhân ma trận truy vấn (query) Q với ma trận khóa (key) K^T để thu được ma trận điểm tương đồng giữa các từ trong câu.
Xét một ví dụ rất nhỏ, trong đó:
$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ và $B = \begin{bmatrix} -1 & 0 \\ 2 & 5 \end{bmatrix}$ là hai ma trận biểu diễn các vector query và key (mỗi hàng tương ứng với một từ, mỗi cột là một chiều đặc trưng).
(a) Tính $C = AB$.
(b) Viết công thức c_{11} dưới dạng tích vô hướng của hàng 1 (A) và cột 1 (B).
(c) Vẽ hình minh họa cho c_{11} và viết code Python.

Lời giải.
(a)

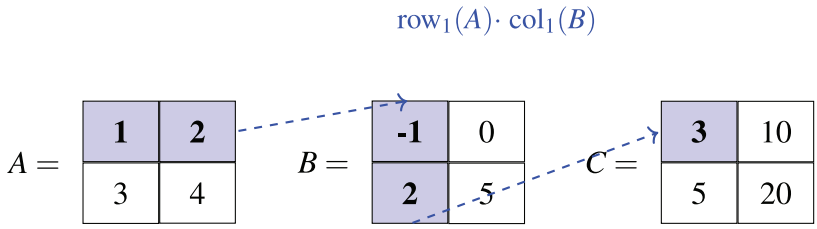
$$C = AB = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} (1\cdot -1 + 2\cdot 2) & (1\cdot 0 + 2\cdot 5) \\ (3\cdot -1 + 4\cdot 2) & (3\cdot 0 + 4\cdot 5) \end{bmatrix} = \begin{bmatrix} 3 & 10 \\ 5 & 20 \end{bmatrix}.$$

(b) $c_{11} = \sum_{k=1}^{2} a_{1k} b_{k1} = a_{11}b_{11} + a_{12}b_{21} = (1)\cdot(-1) + (2)\cdot(2) = 3.$

Đây là tích vô hướng của $\text{row}_1(A) = [1,2]$ và $\text{col}_1(B) = [-1,2]$.

Trong Transformer, phép tính Attention cốt lõi là QK^T. Đây là phép nhân ma trận giữa Query (Q) và Key (K) đã chuyển vị. Mỗi phần tử (i, j) trong ma trận QK^T chính là tích vô hướng (độ tương đồng) giữa vector query của từ thứ i và vector key của từ thứ j.

(c) Minh họa:



Hình 9: Phần tử c_{11} của $C = AB$ là tích vô hướng của hàng 1 (ma trận A) và cột 1 (ma trận B).

(d) Mã Python:

```python
import numpy as np
A = np.array([[1, 2], [3, 4]])
B = np.array([[-1, 0], [2, 5]])

# Dung np.matmul() hoac toan tu @
C = np.matmul(A, B)
# C = A @ B # Tuong duong

print(f"C = A*B =\n{C}")
# [[ 3, 10],
#  [ 5, 20]]
```

Mã lệnh Python 8: Nhân Ma trận với Ma trận (Bài 10)

3 Phương sai, hiệp phương sai, PCA

3.1. Bài tập 11. Phương sai trong AI và NLP

Đề bài. Trong một hệ thống NLP, mỗi câu được ánh xạ thành embedding và ta đo độ liên quan tới chủ đề "AI" bằng cosine similarity giữa embedding câu (đã chuẩn hoá) và vector chủ đề "AI" (đã chuẩn hoá). Khi đó, mỗi câu có một điểm 1 chiều $s_i \in [-1, 1]$.

Giả lập 8 câu, thu được các điểm:

$$s = [0.62,\ 0.58,\ 0.65,\ 0.60,\ 0.10,\ 0.15,\ 0.05,\ 0.12], \quad N = 8.$$

(a) Tính trung bình mẫu $\bar{s}$.
(b) Tính phương sai mẫu (sample variance):

$$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(s_i - \bar{s})^2.$$

(c) Diễn giải ý nghĩa của σ^2 trong ngữ cảnh độ đồng nhất chủ đề của tập câu.

Lời giải.

Phương sai là thước đo quan trọng để lượng hoá độ phân tán thông tin theo một chiều (một feature). Trong PCA, ta tìm các trục mới sao cho phương sai (variance) theo các trục đó được tối đa hoá.

(a) Tính trung bình mẫu.

$$\bar{s} = \frac{1}{N}\sum_{i=1}^{N} s_i = \frac{0.62+0.58+0.65+0.60+0.10+0.15+0.05+0.12}{8} = 0.35875.$$

b Tính phương sai mẫu.
Đặt $d_i = s_i - \bar{s}$. Khi đó:

$$\sum_{i=1}^{N}(s_i-\bar{s})^2 = \sum_{i=1}^{N} d_i^2 = 0.5230875.$$

Suy ra:

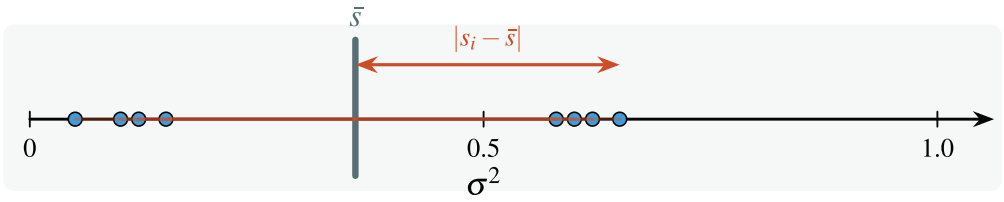
$$\sigma^2 = \frac{0.5230875}{8-1} \approx 0.0747268.$$

Độ lệch chuẩn mẫu:

$$\sigma = \sqrt{\sigma^2} \approx \sqrt{0.0747268} \approx 0.2734.$$

(c) Diễn giải hình học và ý nghĩa NLP.
Các điểm s_i nằm trên trục số (không gian Đề-các (Cartesian) 1D). Trung bình $\bar{s}$ là tâm của tập điểm. Phương sai mẫu đo mức độ phân tán của các điểm quanh $\bar{s}$:

$$\sigma^2 \propto \text{(mức độ toả ra của các điểm quanh tâm).}$$



Hình 10: Minh hoạ: $\bar{s}$ là tâm, các đoạn màu cam biểu diễn độ lệch quanh tâm, σ^2 phản ánh mức độ phân tán.

Trong NLP, σ^2 càng lớn thì mức độ liên quan chủ đề giữa các câu càng không đồng nhất (ví dụ: có thể tồn tại nhiều cụm như rất liên quan và ít liên quan). Ngược lại, σ^2 nhỏ cho thấy các câu có mức liên quan chủ đề tương đối đồng đều.

Mã Python.

```python
import numpy as np

# INPUT: doi du lieu o day
s = np.array([0.62, 0.58, 0.65, 0.60, 0.10, 0.15, 0.05,
   0.12], dtype=float)
N = len(s)

s_mean = s.mean()
dev = s - s_mean
ss = np.sum(dev**2)           # sum of squared deviations
var_sample = ss / (N - 1)     # sample variance (ddof=1)
std_sample = np.sqrt(var_sample)

print("s =", s.tolist())
print("N =", N)
print("mean =", s_mean)
print("sum (s_i-mean)^2 =", ss)
print("sample variance =", var_sample)
print("sample std =", std_sample)
```

Mã lệnh Python 9: Tính trung bình và phương sai mẫu (Bài tập 11)

3.2. Bài tập 12. Hiệp phương sai giữa hai từ (Covariance)

Đề bài. Xét $N = 4$ tài liệu. Gọi x_i là số lần xuất hiện của từ *"AI"* trong tài liệu thứ i, và y_i là số lần xuất hiện của từ *"data"*. Sau khi chuẩn hoá trung bình (center) theo từng từ, ta có:

$$\tilde{x}_i = x_i - \bar{x}, \qquad \tilde{y}_i = y_i - \bar{y},$$

nên $\sum_{i=1}^{N} \tilde{x}_i = 0$ và $\sum_{i=1}^{N} \tilde{y}_i = 0$.

Cho dữ liệu đã trừ trung bình:

$$\tilde{x} = [-3, -1, 1, 3], \qquad \tilde{y} = [-5, -2, 2, 5].$$

(a) Tính hiệp phương sai mẫu (sample covariance) giữa hai từ:

$$\mathrm{Cov}(\tilde{x}, \tilde{y}) = \frac{1}{N-1} \sum_{i=1}^{N} \tilde{x}_i \tilde{y}_i.$$

(b) Vẽ scatter plot của các điểm $(\tilde{x}_i, \tilde{y}_i)$ và giải thích ý nghĩa hình học của dấu (dương/âm) của hiệp phương sai.

(c) Viết mã Python để kiểm chứng kết quả.

Lời giải.

Nhắc lại định nghĩa. Hiệp phương sai mẫu giữa hai biến đã trừ trung bình được định nghĩa bởi:

$$\mathrm{Cov}(\tilde{x},\tilde{y}) = \frac{1}{N-1}\sum_{i=1}^{N}\tilde{x}_i\tilde{y}_i.$$

Vì dữ liệu đã được dịch tâm (center hoá, tức là mỗi giá trị đã được trừ đi trung bình của dãy), công thức trên tương đương với dạng tổng quát:

$$\mathrm{Cov}(x,y) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\bar{x})(y_i-\bar{y}).$$

(a) Tính Hiệp phương sai. Với $N = 4$:

$$\sum_{i=1}^{4}\tilde{x}_i\tilde{y}_i = (-3)(-5)+(-1)(-2)+(1)(2)+(3)(5) = 15+2+2+15 = 34.$$

Do đó:

$$\mathrm{Cov}(\tilde{x},\tilde{y}) = \frac{1}{4-1}\cdot 34 = \frac{34}{3} \approx 11.33.$$

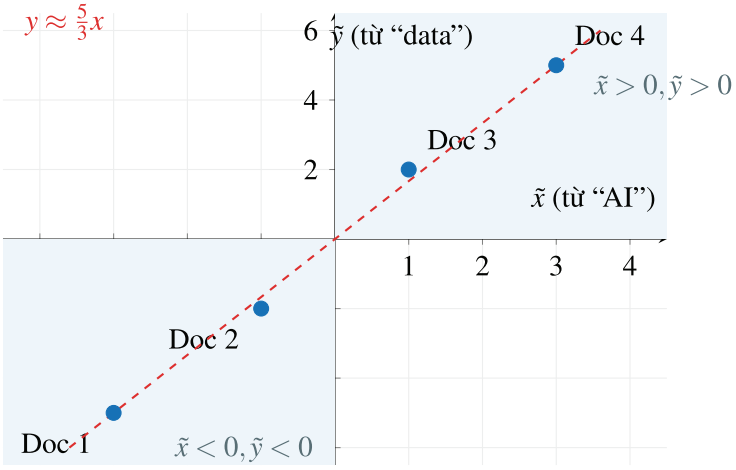
Diễn giải dấu và ý nghĩa.

- Nếu $\tilde{x}_i$ và $\tilde{y}_i$ cùng dấu (cùng dương hoặc cùng âm) thì tích $\tilde{x}_i\tilde{y}_i$ dương $\Rightarrow$ hiệp phương sai dương.
- Nếu $\tilde{x}_i$ và $\tilde{y}_i$ trái dấu thì tích âm $\Rightarrow$ hiệp phương sai âm.

Ở đây các cặp điểm $(\tilde{x}_i,\tilde{y}_i)$ đều có cùng dấu (âm–âm hoặc dương–dương), nên hiệp phương sai là dương và khá lớn.

Hiệp phương sai dương cho thấy, trong các tài liệu mà từ *"AI"* xuất hiện nhiều hơn trung bình ($\tilde{x}_i > 0$) thì từ *"data"* cũng có xu hướng nhiều hơn trung bình ($\tilde{y}_i > 0$), và tương tự cho trường hợp xuất hiện ít hơn trung bình. Đây là một dấu hiệu của đồng biến thiên (co-variation) và gợi ý mối liên hệ đồng xuất hiện.

(b) Hình vẽ Scatter plot.



Hình 11: Scatter plot của $(\tilde{x}_i, \tilde{y}_i)$. Các điểm nằm chủ yếu ở góc phần tư I và III $\Rightarrow \text{Cov}(\tilde{x}, \tilde{y}) > 0$.

Hiệp phương sai là thước đo đồng biến thiên tuyến tính giữa hai biến đã trừ trung bình. Trong bài toán từ vựng, $\text{Cov}(\tilde{x}, \tilde{y}) > 0$ chỉ ra rằng hai từ *"AI"* và *"data"* có xu hướng cùng tăng/giảm so với trung bình trong cùng một tài liệu, gợi ý quan hệ đồng xuất hiện. Lưu ý rằng hiệp phương sai phụ thuộc thang đo (scale) và có thể được chuẩn hoá thành hệ số tương quan để so sánh giữa các cặp từ khác nhau.

(c) Mã Python (minh hoạ và kiểm chứng).

```
import numpy as np

x_tilde = np.array([-3, -1,  1,  3], dtype=float)
y_tilde = np.array([-5, -2,  2,  5], dtype=float)

N = len(x_tilde)

# Cách 1: Theo công  t h c  (N-1)
cov_xy_manual = np.sum(x_tilde * y_tilde) / (N - 1)

# Cách 2: D ng numpy (np.cov chia cho N-1 khi bias=False
    m c  đ nh )
cov_matrix = np.cov(x_tilde, y_tilde, bias=False)
cov_xy_np = cov_matrix[0, 1]

print("Sum x_tilde*y_tilde =", np.sum(x_tilde * y_tilde))
print("Cov(x,y) manual =", cov_xy_manual)
print("Cov matrix:\n", cov_matrix)
print("Cov(x,y) numpy  =", cov_xy_np)
```

Mã lệnh Python 10: Tính hiệp phương sai mẫu (Bài 12)

3.3. Bài tập 13. Ma trận hiệp phương sai (Covariance Matrix)

Đề bài. Xét hai đặc trưng trong ngữ cảnh NLP: số lần xuất hiện của hai từ *"AI"* và *"data"* trong $N = 4$ tài liệu. Gọi x_i (tương ứng từ *"AI"*) và y_i (tương ứng từ *"data"*) là số lần xuất hiện trong tài liệu thứ i. Sau khi dịch tâm (center hoá) theo từng đặc trưng, ta có:

$$\tilde{x}_i = x_i - \bar{x}, \qquad \tilde{y}_i = y_i - \bar{y},$$

và trong đó $i = 1,2,3,4$ lần lượt tương ứng với các tài liệu D_1, D_2, D_3, D_4.

Cho dữ liệu đã dịch tâm:

$$\tilde{x} = [-3, -1, 1, 3], \qquad \tilde{y} = [-5, -2, 2, 5].$$

(Do đã dịch tâm nên trung bình của $\tilde{x}$ và $\tilde{y}$ đều bằng 0.)

(a) Tính $\mathrm{Var}(x)$, $\mathrm{Var}(y)$ và $\mathrm{Cov}(x,y)$ theo mẫu số $N-1$.

(b) Xây dựng ma trận hiệp phương sai $\Sigma \in \mathbb{R}^{2\times 2}$ của vector $\tilde{z}_i = \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \end{bmatrix}$:

$$\Sigma = \begin{bmatrix} \mathrm{Var}(x) & \mathrm{Cov}(x,y) \\ \mathrm{Cov}(y,x) & \mathrm{Var}(y) \end{bmatrix}.$$

(c) Vẽ scatter plot các điểm $(\tilde{x}_i, \tilde{y}_i)$ (gắn nhãn $D_1, \ldots, D_4$) và minh hoạ hướng phân tán chính (trực giác PCA).

(d) Viết mã Python kiểm chứng bằng `numpy`.

Lời giải.

Bước 1: Định nghĩa (mẫu, dịch tâm). Với dữ liệu đã dịch tâm, hiệp phương sai mẫu giữa hai biến được cho bởi

$$\mathrm{Cov}(\tilde{x}, \tilde{y}) = \frac{1}{N-1} \sum_{i=1}^{N} \tilde{x}_i \tilde{y}_i,$$

và phương sai mẫu là trường hợp riêng:

$$\mathrm{Var}(\tilde{x}) = \frac{1}{N-1} \sum_{i=1}^{N} \tilde{x}_i^2, \qquad \mathrm{Var}(\tilde{y}) = \frac{1}{N-1} \sum_{i=1}^{N} \tilde{y}_i^2.$$

Ma trận hiệp phương sai mẫu của $\tilde{z}_i = \begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \end{bmatrix}$ được định nghĩa bởi

$$\Sigma = \frac{1}{N-1} \sum_{i=1}^{N} \tilde{z}_i \tilde{z}_i^{\mathsf{T}}.$$

Khai triển theo từng phần tử:

$$\Sigma = \begin{bmatrix} \frac{1}{N-1}\sum \tilde{x}_i^2 & \frac{1}{N-1}\sum \tilde{x}_i\tilde{y}_i \\ \frac{1}{N-1}\sum \tilde{y}_i\tilde{x}_i & \frac{1}{N-1}\sum \tilde{y}_i^2 \end{bmatrix} = \begin{bmatrix} \mathrm{Var}(x) & \mathrm{Cov}(x,y) \\ \mathrm{Cov}(y,x) & \mathrm{Var}(y) \end{bmatrix}.$$

Do $\tilde{x}_i\tilde{y}_i = \tilde{y}_i\tilde{x}_i$ nên $\mathrm{Cov}(x,y) = \mathrm{Cov}(y,x)$ và Σ là ma trận đối xứng.

Bước 2: Tính toán chi tiết (với $N = 4 \Rightarrow N-1 = 3$).

(a1) Phương sai mẫu của x:

$$\sum_{i=1}^{4} \tilde{x}_i^2 = (-3)^2 + (-1)^2 + (1)^2 + (3)^2 = 9+1+1+9 = 20,$$

$$\mathrm{Var}(x) = \frac{1}{3}\cdot 20 = \frac{20}{3} \approx 6.67.$$

(a2) Phương sai mẫu của y:

$$\sum_{i=1}^{4} \tilde{y}_i^2 = (-5)^2 + (-2)^2 + (2)^2 + (5)^2 = 25+4+4+25 = 58,$$

$$\mathrm{Var}(y) = \frac{1}{3}\cdot 58 = \frac{58}{3} \approx 19.33.$$

(a3) Hiệp phương sai mẫu giữa x và y:

$$\sum_{i=1}^{4} \tilde{x}_i\tilde{y}_i = (-3)(-5) + (-1)(-2) + (1)(2) + (3)(5) = 15+2+2+15 = 34,$$

$$\mathrm{Cov}(x,y) = \frac{1}{3}\cdot 34 = \frac{34}{3} \approx 11.33.$$

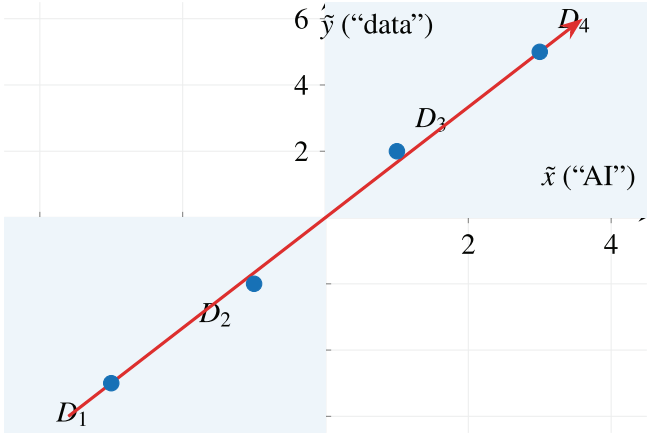
Bước 3: Lập ma trận hiệp phương sai.

$$\Sigma = \begin{bmatrix} \mathrm{Var}(x) & \mathrm{Cov}(x,y) \\ \mathrm{Cov}(y,x) & \mathrm{Var}(y) \end{bmatrix} = \begin{bmatrix} \frac{20}{3} & \frac{34}{3} \\ \frac{34}{3} & \frac{58}{3} \end{bmatrix} \approx \begin{bmatrix} 6.67 & 11.33 \\ 11.33 & 19.33 \end{bmatrix}.$$

Diễn giải

- Các phần tử đường chéo Σ_{11}, Σ_{22} đo mức phân tán riêng của từng đặc trưng.
- Các phần tử ngoài đường chéo $\Sigma_{12} = \Sigma_{21}$ đo đồng biến thiên tuyến tính; giá trị dương lớn cho thấy hai từ có xu hướng cùng tăng/giảm so với trung bình trong cùng tài liệu.
- Trực giác PCA: nếu Σ_{12} lớn, đám mây điểm có xu hướng nằm dọc theo một hướng nghiêng; PCA tìm hướng đó để tối đa hoá phương sai theo trục mới.

(c) Hình minh hoạ (có nhãn $D_1, \ldots, D_4$).



Hình 12: Scatter plot các điểm $(\tilde{x}_i, \tilde{y}_i)$ của 4 tài liệu $D_1, \ldots, D_4$. Đám mây điểm có xu hướng theo hướng nghiêng lên (hiệp phương sai dương), gợi ý hướng phương sai lớn trong PCA.

(d) Mã Python (kiểm chứng).

```
import numpy as np

x_tilde = np.array([-3, -1,  1,  3], dtype=float)
y_tilde = np.array([-5, -2,  2,  5], dtype=float)
N = len(x_tilde)

# Tinh thu cong (chia N-1)
var_x = np.sum(x_tilde**2) / (N - 1)
var_y = np.sum(y_tilde**2) / (N - 1)
cov_xy = np.sum(x_tilde * y_tilde) / (N - 1)

Sigma_manual = np.array([[var_x, cov_xy],
                         [cov_xy, var_y]])

# Numpy: np.cov mac dinh chia N-1 (bias=False)
Sigma_np = np.cov(x_tilde, y_tilde, bias=False)

print("Var(x) =", var_x)
print("Var(y) =", var_y)
print("Cov(x,y) =", cov_xy)
print("\nSigma (manual) =\n", Sigma_manual)
print("\nSigma (np.cov) =\n", Sigma_np)
```

Mã lệnh Python 11: Ma trận hiệp phương sai (Bài 13)

4 Giá trị riêng (Eigenvalue) và Vector riêng (Eigenvector), SVD, LSA

4.1. Bài tập 14. Giá trị riêng và vector riêng

Đề bài. Trong nhiều bài toán NLP, ta biểu diễn mỗi văn bản bằng một vector đặc trưng (feature vector), ví dụ hai trục chủ đề (topic axes) như *"AI"* và *"data"*. Sau khi dịch tâm (center hoá) dữ liệu, ma trận hiệp phương sai 2 chiều thường có dạng đối xứng và nửa xác định dương.

Xét một ma trận hiệp phương sai đơn giản:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

(a) Tìm các giá trị riêng (eigenvalues) λ của A.

(b) Tìm các vector riêng (eigenvectors) $\vec{v}$ tương ứng và chuẩn hoá chúng về độ dài 1.

(c) Minh hoạ hình học: với vector riêng $\vec{v}$, hãy giải thích vì sao $A\vec{v} = \lambda\vec{v}$ nghĩa là A chỉ kéo dãn/thu ngắn theo đúng hướng $\vec{v}$ (không làm đổi phương). Vẽ hình minh hoạ.

(d) Viết mã Python để kiểm chứng.

Lời giải.

Nhắc lại khái niệm. Một số $\lambda \in \mathbb{R}$ được gọi là giá trị riêng của A nếu tồn tại vector khác không $\vec{v} \neq \vec{0}$ sao cho

$$A\vec{v} = \lambda\vec{v}.$$

Khi đó $\vec{v}$ được gọi là vector riêng tương ứng với λ. Các giá trị riêng của A là nghiệm của phương trình đặc trưng

$$\det(A - \lambda I) = 0.$$

(a) Xác định giá trị riêng bằng phương trình đặc trưng. Giá trị riêng λ của A là nghiệm của phương trình:

$$\det(A - \lambda I) = 0,$$

trong đó I là ma trận đơn vị 2×2.

Ta có:

$$A - \lambda I = \begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}.$$

Suy ra:

$$\det(A - \lambda I) = (2-\lambda)(2-\lambda) - 1 = (2-\lambda)^2 - 1.$$

Giải $\det(A-\lambda I)=0$:

$$(2-\lambda)^2-1=0 \iff (2-\lambda)^2=1 \iff 2-\lambda=\pm 1.$$

Do đó:

$$\lambda_1=3, \qquad \lambda_2=1.$$

Kết luận: Hai giá trị riêng của A là $\boxed{\lambda_1=3,\ \lambda_2=1}$.

(b) Tìm vector riêng cho từng giá trị riêng.

(b.1) Với $\lambda_1=3$. Giải hệ:

$$(A-3I)\vec{v}_1=\vec{0}.$$

Ta có:

$$A-3I=\begin{bmatrix}-1 & 1\\ 1 & -1\end{bmatrix}.$$

Gọi $\vec{v}_1=\begin{bmatrix}x\\ y\end{bmatrix}$, khi đó:

$$\begin{bmatrix}-1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}-x+y\\ x-y\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix} \iff -x+y=0 \iff y=x.$$

Vì vậy, $\vec{v}_1$ cùng phương với $\begin{bmatrix}1\\ 1\end{bmatrix}$. Chuẩn hoá về độ dài 1:

$$\vec{v}_1=\frac{1}{\sqrt{1^2+1^2}}\begin{bmatrix}1\\ 1\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ 1\end{bmatrix}.$$

(b.2) Với $\lambda_2=1$. Giải hệ:

$$(A-I)\vec{v}_2=\vec{0}.$$

Ta có:

$$A-I=\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}.$$

Gọi $\vec{v}_2=\begin{bmatrix}x\\ y\end{bmatrix}$, khi đó:

$$\begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix}=\begin{bmatrix}x+y\\ x+y\end{bmatrix}=\begin{bmatrix}0\\ 0\end{bmatrix} \iff x+y=0 \iff y=-x.$$

Vì vậy, $\vec{v}_2$ cùng phương với $\begin{bmatrix}1\\ -1\end{bmatrix}$. Chuẩn hoá về độ dài 1:

$$\vec{v}_2=\frac{1}{\sqrt{1^2+(-1)^2}}\begin{bmatrix}1\\ -1\end{bmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -1\end{bmatrix}.$$

Kết luận:

$$\boxed{\vec{v}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}, \quad \vec{v}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}}$$

(hai vector riêng trực giao và tạo thành một cơ sở trực chuẩn của $\mathbb{R}^2$).

(c) Diễn giải hình học (ý nghĩa của $A\vec{v} = \lambda\vec{v}$). Phương trình

$$A\vec{v} = \lambda\vec{v}$$

nói rằng khi tác động ma trận A lên $\vec{v}$, kết quả không đổi phương (vẫn cùng hướng với $\vec{v}$), chỉ thay đổi độ dài theo hệ số λ:

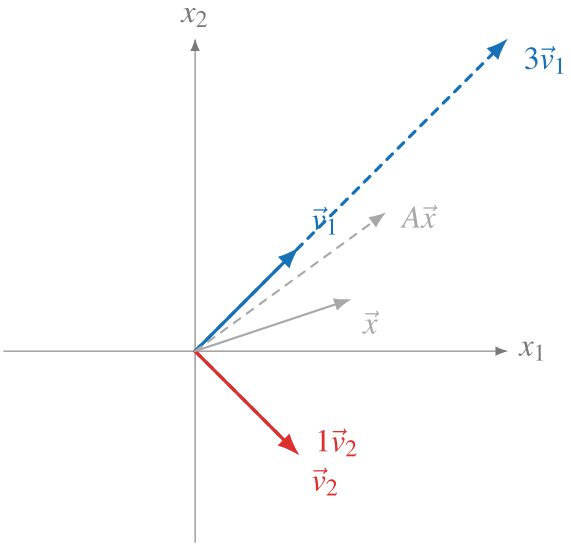
$$\|A\vec{v}\| = |\lambda|\,\|\vec{v}\|.$$

Do đó:

- theo hướng $\vec{v}_1$ dữ liệu bị kéo dãn mạnh với hệ số 3;
- theo hướng $\vec{v}_2$ dữ liệu chỉ kéo dãn nhẹ với hệ số 1.

Trong PCA, các hướng $\vec{v}$ chính là các trục chính (principal directions), còn λ cho biết phương sai (mức thông tin) theo các trục đó.

(c) Hình minh hoạ.



Hình 13: Vector riêng là các hướng bất biến về phương: $A\vec{v}_1 = 3\vec{v}_1$ (kéo dãn mạnh), $A\vec{v}_2 = 1\vec{v}_2$ (giữ nguyên độ dài).

- Ma trận A đối xứng nên có hệ giá trị riêng thực và các vector riêng trực giao.
- Giá trị riêng lớn nhất $\lambda_1 = 3$ xác định hướng phương sai lớn nhất (trục chính thứ nhất) trong PCA.

- Vector riêng tương ứng $\vec{v}_1 = \frac{1}{\sqrt{2}}(1,1)^\top$ là hướng tổng hợp hai trục ban đầu, trong khi $\vec{v}_2 = \frac{1}{\sqrt{2}}(1,-1)^\top$ là hướng tương phản giữa hai trục.

(d) Mã Python.

```python
import numpy as np

A = np.array([[2, 1],
              [1, 2]], dtype=float)

eigvals, eigvecs = np.linalg.eig(A)

print("Eigenvalues:", eigvals)
print("Eigenvectors (columns):\n", eigvecs)

# Kiem chung A v = lambda v cho tung cot vector rieng
for k in range(2):
    v = eigvecs[:, k]
    lam = eigvals[k]
    left = A @ v
    right = lam * v
    print(f"\nCheck k={k}:")
    print("A v =", left)
    print("    v =", right)
```

Mã lệnh Python 12: Tính giá trị riêng và vector riêng (Bài 14)

4.2. Bài tập 15. Phân rã giá trị suy biến(SVD – Singular Value Decomposition)

Đề bài. Trong xử lý ngôn ngữ tự nhiên (NLP), phân rã giá trị suy biến SVD thường được dùng trong Phân tích ngữ nghĩa tiềm ẩn (Latent Semantic Analysis – LSA) để phân tích ma trận thuật ngữ–văn bản (term–document matrix) và rút trích các chủ đề tiềm ẩn (latent topics). Với một ma trận $A \in \mathbb{R}^{m\times n}$, SVD cho phép viết

$$A = U\Sigma V^\top,$$

trong đó U (không gian tài liệu) và V (không gian thuật ngữ) là các ma trận trực giao, còn Σ chứa các giá trị suy biến (singular values) biểu diễn *độ mạnh* của các thành phần tiềm ẩn.

Xét ví dụ rất đơn giản:

$$A = \begin{bmatrix} 3 & 0 \\ 4 & 0 \end{bmatrix} \in \mathbb{R}^{2\times 2}.$$

(a) Thực hiện phân rã SVD của A: $A = U\Sigma V^\top$. Từ đó giải thích vì sao A có hạng 1 và liên hệ trực giác chỉ có một chủ đề tiềm ẩn trong LSA.

(b) Minh hoạ hình học trong $\mathbb{R}^2$: một vector $\vec{x}$ lần lượt chịu tác động của $V^\top$, rồi Σ, rồi U. Vẽ hình minh hoạ.

(c) Viết mã Python để tính SVD và kiểm chứng $A = U\Sigma V^\top$.

Lời giải.

Nhắc lại khái niệm. Với $A \in \mathbb{R}^{m\times n}$, phân rã SVD có dạng

$$A = U\Sigma V^\top,$$

trong đó $U \in \mathbb{R}^{m\times m}$ và $V \in \mathbb{R}^{n\times n}$ là trực giao ($U^\top U = I$, $V^\top V = I$), còn $\Sigma \in \mathbb{R}^{m\times n}$ là ma trận chéo (theo nghĩa tổng quát) với các phần tử không âm $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$. Các σ_i thỏa σ_i^2 là các giá trị riêng của $A^\top A$, và vector riêng tương ứng của $A^\top A$ chính là các cột của V.

(a) Thực hiện SVD của A và giải thích hạng.

Ta tính:

$$A^\top A = \begin{bmatrix} 3 & 4 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 4 & 0 \end{bmatrix} = \begin{bmatrix} 25 & 0 \\ 0 & 0 \end{bmatrix}.$$

Giá trị riêng của $A^\top A$ là 25 và 0, nên các giá trị suy biến là

$$\sigma_1 = \sqrt{25} = 5, \qquad \sigma_2 = \sqrt{0} = 0.$$

Vì chỉ có đúng một giá trị suy biến khác 0 nên

$$\operatorname{rank}(A) = 1.$$

Hai hàng của A là $[3,0]$ và $[4,0]$ cùng phương (hàng thứ hai là $\frac{4}{3}$ lần hàng thứ nhất), nên không tạo được hai chiều độc lập.

Tiếp theo, V lấy từ các vector riêng trực chuẩn của $A^\top A$. Vì $A^\top A$ đã là chéo, ta có thể chọn

$$V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad V^\top = I.$$

Khi đó:

$$\Sigma = \begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix}.$$

Từ $A = U\Sigma V^\top = U\Sigma$, cột thứ nhất của A bằng 5 lần cột thứ nhất của U:

$$A\vec{e}_1 = \begin{bmatrix} 3 \\ 4 \end{bmatrix} = 5\,\vec{u}_1 \implies \vec{u}_1 = \frac{1}{5}\begin{bmatrix} 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 3/5 \\ 4/5 \end{bmatrix}.$$

Chọn $\vec{u}_2$ trực giao với $\vec{u}_1$ và chuẩn hoá, một lựa chọn chuẩn là

$$\vec{u}_2 = \begin{bmatrix} -4/5 \\ 3/5 \end{bmatrix}.$$

Do đó:

$$U = \begin{bmatrix} 3/5 & -4/5 \\ 4/5 & 3/5 \end{bmatrix}.$$

Vậy một phân rã SVD hợp lệ là

$$A = \underbrace{\begin{bmatrix} 3/5 & -4/5 \\ 4/5 & 3/5 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 5 & 0 \\ 0 & 0 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}_{V^\top}.$$

Trong LSA, A thường là ma trận TF–IDF (thuật ngữ $\times$ văn bản). Nếu A có hạng 1, điều đó cho thấy dữ liệu gần như bị chi phối bởi một hướng tiềm ẩn duy nhất. Theo trực giác, chỉ có một *chủ đề* nổi trội giải thích cấu trúc đồng xuất hiện trong ma trận.

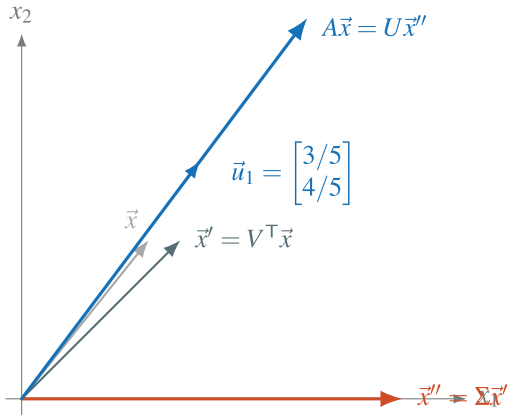
(b) Minh hoạ hình học của $A\vec{x} = U\Sigma V^\top \vec{x}$.

Ta có thể hiểu theo ba bước:

$$\vec{x} \xrightarrow{V^\top} \vec{x}' \xrightarrow{\Sigma} \vec{x}'' \xrightarrow{U} A\vec{x}.$$

Trong ví dụ này $V^\top = I$ nên bước đầu không đổi. Ma trận Σ gây giảm chiều theo trục thứ hai (nhân 0) và co giãn theo trục thứ nhất (nhân 5). Cuối cùng, U là phép quay/đổi cơ sở trực giao đưa trục suy biến về hướng của $\begin{bmatrix} 3 \\ 4 \end{bmatrix}$.

(b) Hình minh hoạ.



Hình 14: Ý nghĩa hình học của SVD: $V^\top$ đổi cơ sở, Σ co giãn theo các trục suy biến (ở đây trục 2 bị triệt tiêu), và U quay/đổi cơ sở đưa kết quả về không gian gốc.

(c) Mã Python.

```python
import numpy as np

A = np.array([[3.0, 0.0],
              [4.0, 0.0]])

# full_matrices=True: U (2x2), VT (2x2)
U, s, VT = np.linalg.svd(A, full_matrices=True)

# Lap Sigma tu vector singular values s
Sigma = np.zeros_like(A)
Sigma[np.diag_indices(min(A.shape))] = s

print("A =\n", A)
print("\nU =\n", U)
print("\nsingular values =", s)
print("\nSigma =\n", Sigma)
print("\nVT =\n", VT)

A_recon = U @ Sigma @ VT
print("\nU @ Sigma @ VT =\n", A_recon)
```

Mã lệnh Python 13: Tính SVD và kiểm chứng (Bài 15)

4.3. Bài tập 16. PCA/LSA thu nhỏ (mini) với SVD rút ngọn (TruncatedSVD)

Đề bài. Trong một hệ thống NLP đơn giản, ta có $N = 6$ câu tiếng Việt thuộc ba nhóm chủ đề: Công nghệ thông tin (CNTT), du lịch và AI/Dữ liệu. Cho danh sách câu:

```python
docs = [
    "sinh vien VKU hoc cong nghe thong tin",        # CNTT
    "truong VKU dao tao ky su cong nghe so",        # CNTT
    "du lich Da Nang rat phat trien",               # Du lich
    "bien My Khe va Son Tra noi tieng",             # Du lich
    "tri tue nhan tao AI va du lieu lon",           # AI/Data
    "hoc may machine learning va deep learning"     # AI/Data
]
```

Ta muốn biểu diễn mỗi câu bằng vector TF–IDF, thu được ma trận $X \in \mathbb{R}^{N\times d}$, và đồng thời giảm chiều X xuống không gian $k = 2$ bằng `TruncatedSVD` để quan sát cấu trúc chủ đề tiềm ẩn (một LSA/PCA mini).

(a) Vector hoá 6 câu bằng TF–IDF để thu được $X \in \mathbb{R}^{6\times d}$.

(b) Giảm chiều xuống $k = 2$ bằng `TruncatedSVD` để thu được $X_2 \in \mathbb{R}^{6\times 2}$.

(c) In toạ độ 2D của 6 câu và vẽ hình trực quan để quan sát xu hướng gom cụm.

Lời giải.

Nhắc lại khái niệm. TF–IDF là một cách gán trọng số cho từ trong văn bản dựa trên hai yếu tố: tần suất xuất hiện của từ trong chính văn bản (TF) và mức độ hiếm của từ trong toàn bộ tập văn bản (IDF). Trong thực hành, `TfidfVectorizer` tạo ra ma trận TF–IDF thưa $X \in \mathbb{R}^{N\times d}$, với N là số văn bản và d là kích thước từ vựng. Cụ thể, với tập N văn bản, TF–IDF gán trọng số cho thuật ngữ t trong văn bản d dựa trên: tần suất xuất hiện (TF) và mức độ hiếm (IDF). Một dạng IDF thường dùng:

$$\mathrm{idf}(t) = \log \frac{N}{\mathrm{df}(t)},$$

trong đó $\mathrm{df}(t)$ là số văn bản chứa t. Sau TF–IDF ta nhận được ma trận thưa $X \in \mathbb{R}^{N\times d}$.

`TruncatedSVD` tìm xấp xỉ hạng thấp của X theo thuật toán SVD:

$$X \approx U_k \Sigma_k V_k^{\mathsf{T}},$$

trong đó k là số thành phần giữ lại. Khi $k = 2$, mỗi văn bản được gán một điểm trong $\mathbb{R}^2$, và ta có thể trực quan hoá để quan sát cấu trúc chủ đề tiềm ẩn (LSA thu nhỏ).

(a) Lập ma trận TF–IDF $X \in \mathbb{R}^{6\times d}$.

Vì $N = 6$, sau khi xây dựng từ vựng, ta thu được kích thước d (số từ khác nhau). Mỗi câu tương ứng một hàng của X, mỗi cột tương ứng một thuật ngữ trong từ vựng.

Để minh hoạ cách hình thành trọng số, xét hai từ đại diện:

- Từ ‘‘`VKU`’’ xuất hiện trong đúng hai câu CNTT (câu 1 và 2), nên

$$\mathrm{df}(\texttt{VKU}) = 2 \quad \Rightarrow \quad \mathrm{idf}(\texttt{VKU}) = \log \frac{6}{2} = \log 3.$$

- Từ ‘‘`My`’’ (trong cụm `My Khe`) chỉ xuất hiện trong đúng một câu du lịch (câu 4), nên

$$\mathrm{df}(\texttt{My}) = 1 \quad \Rightarrow \quad \mathrm{idf}(\texttt{My}) = \log \frac{6}{1} = \log 6.$$

Do đó, các từ hiếm (ví dụ `My`) thường có IDF lớn và mang tính phân biệt chủ đề cao hơn, còn các từ xuất hiện trong nhiều câu sẽ có IDF nhỏ hơn.

Kết quả của bước này là một ma trận thưa

$$X = \begin{bmatrix} — \text{ TF–IDF của câu 1 } — \\ — \text{ TF–IDF của câu 2 } — \\ \vdots \\ — \text{ TF–IDF của câu 6 } — \end{bmatrix} \in \mathbb{R}^{6\times d}.$$

(b) Giảm chiều bằng `TruncatedSVD` xuống $k = 2$.

Xét phân rã suy biến (cắt ngọn) của X:

$$X \approx U_2 \Sigma_2 V_2^{\mathsf{T}}, \qquad U_2 \in \mathbb{R}^{6\times 2},\ \Sigma_2 \in \mathbb{R}^{2\times 2},\ V_2 \in \mathbb{R}^{d\times 2}.$$

Trong ngữ cảnh LSA:

- mỗi hàng của U_2 (hoặc chính xác hơn là $U_2\Sigma_2$) biểu diễn một câu trong không gian chủ đề 2 chiều;
- mỗi cột của V_2 mô tả một trục chủ đề trong không gian thuật ngữ;
- các phần tử đường chéo của Σ_2 là độ mạnh của hai thành phần tiềm ẩn quan trọng nhất.

Trong `scikit-learn`, `TruncatedSVD.fit_transform(X)` trả về toạ độ:

$$X_2 = XV_2 \in \mathbb{R}^{6\times 2}.$$

Vì vậy, hàng thứ i của X_2 chính là toạ độ 2D của câu i trong không gian tiềm ẩn.

(c) In toạ độ 2 chiều và trực quan hoá.

Gán nhãn cho 6 câu:

$$\text{CNTT1, CNTT2, DL1, DL2, AI1, AI2.}$$

Khi thu được

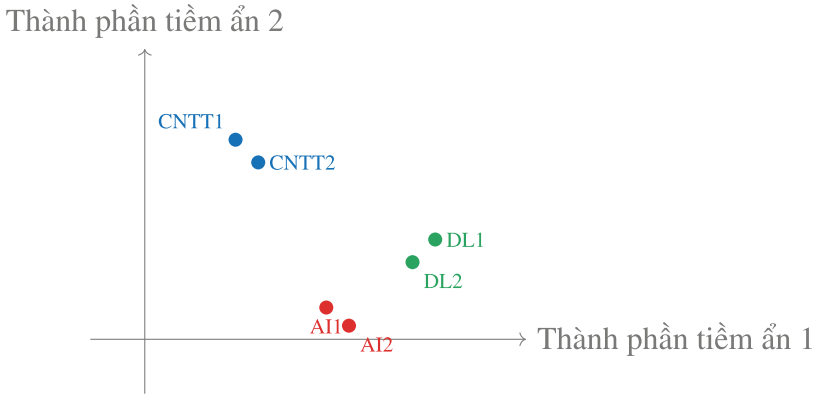
$$X_2 = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_6 & y_6 \end{bmatrix},$$

ta vẽ 6 điểm (x_i, y_i) trên mặt phẳng.

Nếu mô hình LSA thu nhỏ rút trích tốt cấu trúc ngữ nghĩa tiềm ẩn, ta kỳ vọng:

- CNTT1 gần CNTT2;
- DL1 gần DL2;
- AI1 gần AI2;
- ba cụm này tách tương đối khỏi nhau.

(c) Hình minh hoạ giả lập.



Hình 15: Minh hoạ trực giác: sau TF–IDF và SVD cắt ngọn xuống 2D, các câu cùng chủ đề thường nằm gần nhau hơn trong không gian tiềm ẩn.

Trong bài toán này, X là ma trận TF–IDF của 6 câu. Khi áp dụng `TruncatedSVD` với $k = 2$, ta thu được $X_2 \in \mathbb{R}^{6\times 2}$, cung cấp biểu diễn 2D cho từng câu. Trực quan hoá thường cho thấy xu hướng gom cụm theo ba nhóm chủ đề (CNTT, du lịch, AI/Dữ liệu), qua đó minh hoạ trực giác nền tảng của LSA: rút trích cấu trúc ngữ nghĩa tiềm ẩn bằng xấp xỉ hạng thấp.

Mã Python (đặt cuối, dùng để tính số liệu và vẽ hình).

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import
   TfidfVectorizer
from sklearn.decomposition import TruncatedSVD

docs = [
    "sinh vien VKU hoc cong nghe thong tin",
    "truong VKU dao tao ky su cong nghe so",
    "du lich Da Nang rat phat trien",
    "bien My Khe va Son Tra noi tieng",
    "tri tue nhan tao AI va du lieu lon",
    "hoc may machine learning va deep learning"
]
labels = ["CNTT1", "CNTT2", "DL1", "DL2", "AI1", "AI2"]

# (a) TF--IDF
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(docs)
print("Kich thuoc TF-IDF X (N, d) =", X.shape)
print("So tu vung d =", X.shape[1])

# (b) TruncatedSVD (k=2)
```

```
svd = TruncatedSVD(n_components=2, random_state=42)
X_2 = svd.fit_transform(X)

print("\nExplained variance ratio:", svd.
   explained_variance_ratio_)
print("Tong ty le:", svd.explained_variance_ratio_.sum())

# (c) In toa do 2D
print("\nToa do 2D:")
for i, doc in enumerate(docs):
    x, y = X_2[i]
    print(f"{labels[i]}: ({x:.3f}, {y:.3f}) -> {doc}")

# Ve scatter plot
xs = X_2[:, 0]
ys = X_2[:, 1]

plt.figure(figsize=(6, 5))
plt.scatter(xs, ys)

for i, lab in enumerate(labels):
    plt.text(xs[i] + 0.01, ys[i] + 0.01, lab)

plt.axhline(0, linewidth=0.6)
plt.axvline(0, linewidth=0.6)
plt.xlabel("Thanh phan tiem an 1")
plt.ylabel("Thanh phan tiem an 2")
plt.title("TF-IDF + TruncatedSVD: Bieu dien 6 cau trong
   khong gian 2D")
plt.tight_layout()
plt.show()
```

Mã lệnh Python 14: TF–IDF + TruncatedSVD k=2 (Bài 16)

5 Các phép toán với Tensor, Convolution, Attention

5.1. Bài tập 17. Hoán vị trục (Permute) trong Multi-Head Attention

Đề bài. Trong Multi-Head Attention, đầu ra của các head thường được lưu dưới dạng tensor

$$\mathscr{T}_{\text{heads}} \in \mathbb{R}^{B\times h\times S\times d_k},$$

trong đó B là batch size, h là số head, S là độ dài chuỗi (sequence length), và d_k là số chiều của mỗi head. Để chuẩn bị cho bước gộp head, ta cần hoán vị trục để đưa

tensor về dạng

$$\mathscr{T}_{\text{perm}} \in \mathbb{R}^{B\times S\times h\times d_k}.$$

Với $B=2$, $h=8$, $S=10$, $d_k=64$, hãy:

(a) Xác định phép hoán vị trục cần dùng và suy ra kích thước tensor sau hoán vị.

(b) Minh hoạ bằng hình vẽ và nêu rõ vì sao cần hoán vị trước khi gộp head.

Lời giải.

Nhắc lại khái niệm. Với tensor T bậc 4 có kích thước (B,h,S,d_k), phép `permute` chỉ thay đổi thứ tự trục (axis order) mà không thay đổi các phần tử. Nói cách khác, `permute` chỉ tạo một lượt nhìn (view) với cách diễn giải chỉ số khác. Nếu cần đổi thứ tự từ $(0,1,2,3)$ sang $(0,2,1,3)$ thì ta viết:

$$T_{\text{perm}} = T.\text{permute}(0,2,1,3).$$

(a) Phép hoán vị trục và kích thước sau hoán vị.

Tensor ban đầu:

$$\mathscr{T}_{\text{heads}} \in \mathbb{R}^{B\times h\times S\times d_k}.$$

Mục tiêu là đổi thứ tự trục *head* và *sequence* để hai trục (h,d_k) trở thành liền kề ở cuối tensor:

$$(B,h,S,d_k) \longrightarrow (B,S,h,d_k).$$

Do đó phép hoán vị trục cần dùng là

$$\mathscr{T}_{\text{perm}} = \mathscr{T}_{\text{heads}}.\text{permute}(0,2,1,3).$$

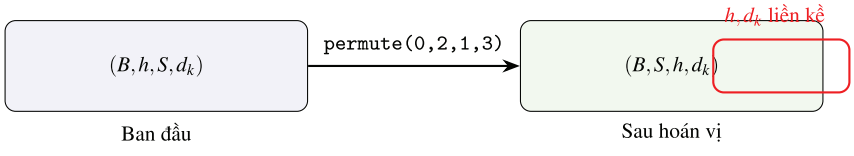
Với $B=2$, $h=8$, $S=10$, $d_k=64$:

$$(2,8,10,64) \longrightarrow (2,10,8,64).$$

(b) Diễn giải vì sao cần hoán vị trước khi gộp head.

Mục tiêu cuối cùng (bài sau) là gộp h và d_k thành $E = h\cdot d_k$ để thu tensor kích thước (B,S,E). Tuy nhiên, trong tensor ban đầu (B,h,S,d_k) thì h và d_k *không liền kề* (do S nằm giữa), nên không thể gộp trực tiếp bằng một phép `view/reshape` tuyến tính theo layout bộ nhớ. Sau khi `permute` về (B,S,h,d_k), hai trục cuối (h,d_k) trở thành liền kề, khi đó mới có thể gộp.

Hình minh hoạ.



Hình 16: Bước hoán vị trục: $(B,h,S,d_k) \to (B,S,h,d_k)$ để đưa hai trục (h,d_k) về liền kề trước khi gộp.

Phép `permute(0,2,1,3)` là thao tác chuẩn trong Transformer nhằm đặt chiều chuỗi S đứng ngay sau batch B, đồng thời đưa (h, d_k) về cuối để chuẩn bị gộp head.

5.2. Bài tập 18. Định hình lại (Reshape/View) để gộp head trong Multi-Head Attention

Đề bài. Tiếp tục từ Bài 17, sau khi hoán vị ta có tensor

$$\mathscr{T}_{\text{perm}} \in \mathbb{R}^{B\times S\times h\times d_k}.$$

Ta muốn gộp hai trục cuối (h, d_k) thành một trục embedding duy nhất

$$E = h \cdot d_k, \qquad \mathscr{T}_{\text{final}} \in \mathbb{R}^{B\times S\times E}.$$

Với $B = 2$, $h = 8$, $S = 10$, $d_k = 64$, hãy:

(a) Tính E và suy ra kích thước mong muốn của $\mathscr{T}_{\text{final}}$.

(b) Trình bày phép `reshape`/`view` phù hợp và giải thích yêu cầu vùng nhớ liền kề (contiguous()) khi dùng `view`.

(c) Viết mã PyTorch minh hoạ và kiểm chứng kích thước các tensor.

Lời giải.

Nhắc lại khái niệm. `reshape` và `view` là các phép *định hình lại* tensor (thay đổi shape) nhưng không thay đổi dữ liệu.

(a) Tính E và kích thước đích.

Vì $E = h \cdot d_k$, với $h = 8$, $d_k = 64$:

$$E = 8 \cdot 64 = 512.$$

Do đó:

$$(B, S, h, d_k) = (2, 10, 8, 64) \longrightarrow (B, S, E) = (2, 10, 512).$$

(b) Phép gộp hai trục cuối.

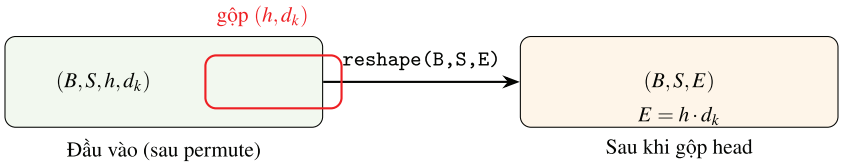
Từ $\mathscr{T}_{\text{perm}} \in \mathbb{R}^{B\times S\times h\times d_k}$, ta gộp (h, d_k) thành E:

$$\mathscr{T}_{\text{final}} = \text{reshape}(\mathscr{T}_{\text{perm}}, B, S, E).$$

Nếu dùng `view` (nhanh và không copy) thì cần bảo đảm contiguous:

$$\mathscr{T}_{\text{final}} = \mathscr{T}_{\text{perm}}.\text{contiguous}().\text{view}(B, S, E).$$

Hình minh hoạ.



Hình 17: Bước gộp head: $(B,S,h,d_k) \to (B,S,E)$ với $E = h \cdot d_k$.

Sau khi gộp head, tensor thu được có kích thước (B,S,E) (embedding chiều E cho mỗi vị trí trong chuỗi). Đây là định dạng chuẩn để đưa qua lớp tuyến tính cuối của attention và/hoặc khối Feed-Forward trong Transformer.

Mã PyTorch (đặt cuối, minh hoạ và kiểm chứng).

```
import torch

B, h, S, d_k = 2, 8, 10, 64
E = h * d_k

T_heads = torch.randn(B, h, S, d_k)
print("Input (B,h,S,d_k):", tuple(T_heads.shape))

# Bài 17: permute -> (B,S,h,d_k)
T_perm = T_heads.permute(0, 2, 1, 3)
print("After permute (B,S,h,d_k):", tuple(T_perm.shape))

# Bài 18: gop (h,d_k) -> E
T_final_view = T_perm.contiguous().view(B, S, E)
print("After view (B,S,E):", tuple(T_final_view.shape))

# reshape (tuong duong, thuong an toan hon)
T_final_reshape = T_perm.reshape(B, S, E)
print("After reshape (B,S,E):", tuple(T_final_reshape.
    shape))
```

Mã lệnh Python 15: Permute va Reshape/View de gom head (Bài 17–18)

5.3. Bài tập 19. Tích chập 1D trong TextCNN cho NLP

Đề bài. Trong các mô hình *TextCNN* dùng cho xử lý ngôn ngữ tự nhiên (NLP), tầng tích chập 1D (Conv1D) thường được áp dụng trên chuỗi đặc trưng của câu, ví dụ chuỗi embedding theo chiều thời gian (theo vị trí từ).

Giả sử một câu rất ngắn được mã hoá thành chuỗi đặc trưng 1D

$$x = [1,2,3,4] \quad (L = 4),$$

trong đó mỗi phần tử biểu diễn mức kích hoạt của một đặc trưng tại vị trí từ tương ứng.

Ta sử dụng một bộ lọc (kernel) 1D

$$w = [1, 0, -1] \quad (k = 3),$$

với *valid padding* (không đệm biên, tức là không thêm phần tử 0 ở hai đầu) và bước trượt (stride) $= 1$.

Hãy tính đầu ra của phép tích chập 1D:

$$y = x * w.$$

Công thức (theo dạng tương quan chéo thường dùng trong Conv1D):

$$y_i = \sum_{j=0}^{k-1} w_j x_{i+j}, \qquad i = 0, 1, \ldots, L-k.$$

Lời giải.

Nhắc lại khái niệm. Với đầu vào 1D độ dài L, kernel độ dài k, bước trượt $s = 1$ và *valid padding*, độ dài đầu ra là

$$L_{\text{out}} = L - k + 1.$$

Mỗi phần tử y_i được tính bằng tích vô hướng giữa kernel w và đoạn con độ dài k của x bắt đầu tại vị trí i.

(a) Xác định độ dài đầu ra.

Vì $L = 4$, $k = 3$ nên

$$L_{\text{out}} = 4 - 3 + 1 = 2.$$

Do đó y có hai phần tử: $y = [y_0, y_1]$.

(b) Tính các giá trị y_0, y_1.

Tại vị trí $i = 0$: cửa sổ con là $[x_0, x_1, x_2] = [1, 2, 3]$,

$$y_0 = w_0 x_0 + w_1 x_1 + w_2 x_2 = 1 \cdot 1 + 0 \cdot 2 + (-1) \cdot 3 = 1 - 3 = -2.$$

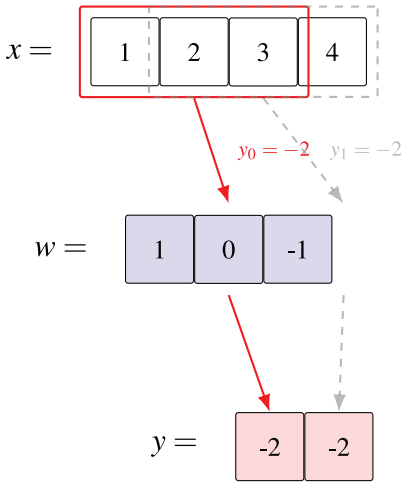
Tại vị trí $i = 1$: cửa sổ con là $[x_1, x_2, x_3] = [2, 3, 4]$,

$$y_1 = w_0 x_1 + w_1 x_2 + w_2 x_3 = 1 \cdot 2 + 0 \cdot 3 + (-1) \cdot 4 = 2 - 4 = -2.$$

Vậy kết quả cuối cùng:

$$\boxed{y = [-2, -2].}$$

Hình minh hoạ.



Hình 18: Minh hoạ tích chập 1D (Conv1D): kernel w trượt qua x với bước trượt 1 và tính tích vô hướng tại mỗi vị trí (valid padding).

Kernel $w = [1, 0, -1]$ đo chênh lệch giữa hai đầu của mỗi cửa sổ độ dài là 3 (một dạng đạo hàm rời rạc), nên có thể xem như một bộ phát hiện biến thiên (edge-like) trong chuỗi. Trong TextCNN, các kernel được học từ dữ liệu và đóng vai trò như các bộ phát hiện mẫu n-gram (ở đây là 3-gram) trên chuỗi đặc trưng.

Mã Python (đặt cuối, minh hoạ và kiểm chứng).

```
import numpy as np

x = np.array([1, 2, 3, 4], dtype=float)
w = np.array([1, 0, -1], dtype=float)

L = len(x)
k = len(w)
L_out = L - k + 1

y = np.zeros(L_out, dtype=float)
for i in range(L_out):
    y[i] = np.sum(w * x[i:i+k])

print("x =", x.tolist())
print("w =", w.tolist())
print("y =", y.tolist())   # [-2.0, -2.0]
```

Mã lệnh Python 16: Tich chap 1D (valid, stride=1) (Bài 19)

5.4. Bài tập 20. TextCNN thu nhỏ: tích chập trên embedding và max-pooling theo thời gian

Đề bài. Xét ma trận embedding của một câu gồm $L = 4$ từ, mỗi từ có embedding chiều $d = 3$:

$$X_{\text{emb}} \in \mathbb{R}^{4\times 3}, \qquad X_{\text{emb}} = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

Xét một bộ lọc (kernel) của TextCNN có bề rộng đúng bằng chiều embedding và chiều cao $k = 2$:

$$w \in \mathbb{R}^{2\times 3}, \qquad w = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \end{bmatrix}.$$

Ta thực hiện tích chập theo trục thời gian (bước trượt $= 1$, *valid padding* tức là không đệm biên).

(a) Tính đầu ra sau tích chập (feature map C) và nêu kích thước của C.

(b) Áp dụng lấy max theo trục L của C (max-pooling) để thu được $\hat{c}$.

Lời giải.

Nhắc lại khái niệm. Trong TextCNN, với một kernel $w \in \mathbb{R}^{k\times d}$ trượt trên $X_{\text{emb}} \in \mathbb{R}^{L\times d}$ (stride 1, không đệm biên), tại vị trí i ta lấy cửa sổ con

$$X_{(i)} = X_{\text{emb}}[i : i+k,\ :] \in \mathbb{R}^{k\times d},$$

và tính một giá trị vô hướng (tích vô hướng theo phần tử rồi cộng):

$$c_i = \langle w, X_{(i)} \rangle = \sum_{a=1}^{k} \sum_{b=1}^{d} w_{ab} X_{(i),ab}, \qquad i = 1, 2, \ldots, L-k+1.$$

Do đó $C = [c_1, \ldots, c_{L-k+1}]^{\mathsf{T}} \in \mathbb{R}^{(L-k+1)\times 1}$. Sau đó lấy max (max-pooling) theo thời gian trả về

$$\hat{c} = \max_i c_i,$$

tức là lấy kích hoạt lớn nhất của filter trên toàn câu.

(a) Tính đầu ra sau tích chập (feature map) C.

Vì $L = 4$, $k = 2$ và bước trượt $= 1$ (không đệm biên), số vị trí trượt là

$$L_{\text{out}} = L - k + 1 = 4 - 2 + 1 = 3.$$

Vì vậy $C \in \mathbb{R}^{3\times 1}$ với $C = [c_1, c_2, c_3]^{\mathsf{T}}$.

Vị trí $i = 1$: cửa sổ con là hai hàng đầu của X_{emb}:

$$X_{(1)} = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}.$$

Khi đó

$$\begin{aligned} c_1 &= \sum_{a=1}^{2} \sum_{b=1}^{3} w_{ab} X_{(1),ab} \\ &= (1\cdot 1 + 0\cdot 0 + 0\cdot 1) + ((-1)\cdot 2 + 1\cdot 1 + 0\cdot 0) \\ &= 1 + (-2+1) = 0. \end{aligned}$$

Vị trí $i = 2$: cửa sổ con là hai hàng 2–3:

$$X_{(2)} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 0 \end{bmatrix}.$$

Khi đó

$$\begin{aligned} c_2 &= (1\cdot 2 + 0\cdot 1 + 0\cdot 0) + ((-1)\cdot 1 + 1\cdot 3 + 0\cdot 0) \\ &= 2 + (-1+3) = 4. \end{aligned}$$

Vị trí $i = 3$: cửa sổ con là hai hàng 3–4:

$$X_{(3)} = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

Khi đó

$$\begin{aligned} c_3 &= (1\cdot 1 + 0\cdot 3 + 0\cdot 0) + ((-1)\cdot 0 + 1\cdot 1 + 0\cdot 1) \\ &= 1 + (0+1) = 2. \end{aligned}$$

Vậy

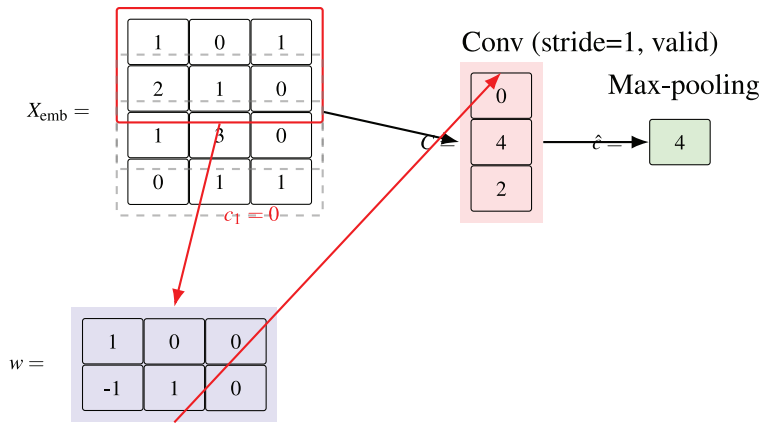
$$\boxed{C = \begin{bmatrix} 0 \\ 4 \\ 2 \end{bmatrix} \in \mathbb{R}^{3\times 1}.}$$

(b) Max-pooling theo thời gian.

Max-pooling theo thời gian lấy giá trị lớn nhất của C:

$$\boxed{\hat{c} = \max(C) = \max(0,4,2) = 4.}$$

Hình minh hoạ.



Hình 19: TextCNN thu nhỏ: kernel $w \in \mathbb{R}^{2\times 3}$ trượt theo trục thời gian trên $X_{\text{emb}} \in \mathbb{R}^{4\times 3}$ để tạo feature map $C \in \mathbb{R}^{3\times 1}$, sau đó max-pooling theo thời gian cho ra $\hat{c} = 4$.

Nhận xét. Trong TextCNN thực tế, ta dùng nhiều kernel với các kích thước k khác nhau (ví dụ $k = 2,3,4,\ldots$) và nhiều filter cho mỗi k. Mỗi filter cho một giá trị pooled $\hat{c}$; ghép các $\hat{c}$ lại ta thu được vector đặc trưng mức câu (sentence-level feature vector). Max-pooling theo thời gian đóng vai trò chọn kích hoạt mạnh nhất của mẫu n-gram mà filter đã học, bất kể vị trí xuất hiện trong câu.

Mã Python (đặt cuối, minh hoạ và kiểm chứng).

```
import numpy as np

X = np.array([
    [1, 0, 1],
    [2, 1, 0],
    [1, 3, 0],
    [0, 1, 1]
], dtype=float)

w = np.array([
    [1, 0, 0],
    [-1, 1, 0]
], dtype=float)

L, d = X.shape
k = w.shape[0]
L_out = L - k + 1

C = np.zeros(L_out, dtype=float)
for i in range(L_out):
    window = X[i:i+k, :]
    C[i] = np.sum(w * window)
```

```
chat = np.max(C)

print("C =", C.tolist())          # [0.0, 4.0, 2.0]
print("c_hat =", float(chat))     # 4.0
```

Mã lệnh Python 17: TextCNN mini: conv + max-pooling (Bài 20)

Kết luận

Hệ thống bài tập trong chương này đã củng cố và mở rộng các khái niệm của Đại số tuyến tính trong bối cảnh Trí tuệ nhân tạo và Xử lý ngôn ngữ tự nhiên. Từ các phương pháp biểu diễn dữ liệu văn bản (one-hot, Bag-of-Words, TF–IDF, embedding) đến các phép toán cốt lõi trên vector, ma trận và tensor, bạn đọc đã thực hành nhiều kỹ thuật nền tảng như đo tương đồng cosine, biến đổi tuyến tính, hiệp phương sai và phân rã (eigen, SVD/LSA), cũng như các phép toán đặc trưng của mô hình học sâu (tích chập, attention, thao tác permute/reshape trong multi-head attention).

Việc nắm vững các cấu trúc và phép toán Đại số tuyến tính không chỉ giúp hiểu rõ cơ chế hình học và đại số phía sau các mô hình hiện đại (TextCNN, Transformer), mà còn tạo nền tảng phương pháp luận để triển khai, phân tích và tối ưu các bài toán NLP thực tiễn như biểu diễn ngữ nghĩa, tìm kiếm tương đồng, giảm chiều, phát hiện chủ đề và trích chọn đặc trưng.

Chương 3

XÂY DỰNG CÁC HỆ THỐNG AI VÀ NLP

ỨNG DỤNG ĐẠI SỐ TUYẾN TÍNH

Giới thiệu 83
1 Xây dựng hệ thống AI tìm kiếm tài liệu tương đồng 83
1.1. Đề bài mô tả yêu cầu 83
1.2. Quy trình tổng quát 84
Bước 1: Tiền xử lý và xây dựng từ vựng 84
Bước 2: Tính TF, DF, IDF và TF–IDF 85
Bước 3: Biểu diễn mỗi tài liệu dưới dạng vector 88
Bước 4: Tính Cosine Similarity và giải thích 90
Bước 5: Xếp hạng và lựa chọn Top-k 91
1.3. Trực quan hoá trong không gian vector 93
1.4. Kết luận và mở rộng bài toán 94
1.5. Mã Python (kèm chú thích chi tiết) 94
2 Xây dựng hệ thống AI phân cụm văn bản 100
2.1. Đề bài mô tả yêu cầu 100
2.2. Quy trình tổng quát 100
Bước 1: Tiền xử lý văn bản 100
Bước 2: Word2Vec – Biểu diễn từ dưới dạng vector 101
Bước 3: Giảm chiều bằng SVD 105
Bước 4: Phân cụm bằng K-Means 107
2.3. Trực quan hoá phân cụm 2D 113
2.4. Kết luận và mở rộng bài toán 114
2.5. Mã Python (kèm chú thích chi tiết) 114
3 Xây dựng hệ thống AI xác định và nhóm tài liệu theo chủ đề 119
3.1. Đề bài mô tả yêu cầu 119
3.2. Quy trình tổng quát 119
Bước 1: Xây dựng ma trận TF–IDF 121
Bước 2: Phân rã SVD 122
Bước 3: Trích xuất và xác định chủ đề 123
Bước 4: Liên kết tài liệu với chủ đề (Document Clustering) 125
Bước 5: Minh họa chủ đề và Không gian ngữ nghĩa (Semantic Space Visualization) 126
3.3. Kết luận và mở rộng bài toán 127
3.4. Mã Python (kèm chú thích chi tiết) 128
Kết luận 135

Giới thiệu

Sau khi đã tiếp cận lý thuyết và ứng dụng đại số tuyến tính để giải quyết các bài toán xử lý tác vụ cơ bản trong trí tuệ nhân tạo và xử lý ngôn ngữ tự nhiên ở các chương trước, nội dung chương này sẽ giúp người học triển khai từng bước cụ thể của quy trình xây dựng các dự án AI và NLP trong thực tế gồm:

- **Xây dựng hệ thống AI tìm kiếm tài liệu tương đồng** dựa vào các kỹ thuật TF–IDF, tính độ tương đồng cosine để xếp hạng các tài liệu dựa theo mức độ liên quan, trực quan hoá kết quả dưới dạng các vector trong không gian vector.
- **Xây dựng hệ thống AI phân cụm văn bản** sử dụng Word2Vec (semantic embedding), SVD/Truncated SVD để giảm chiều vector xuống không gian 2D, K-Means để phân cụm tài liệu, trực quan hoá phân cụm trong không gian 2D.
- **Xây dựng hệ thống AI xác định và nhóm tài liệu theo chủ đề** sử dụng TF–IDF, LSA (SVD) để giảm chiều, trích rút, xác định và phân nhóm tài liệu theo chủ đề, trực quan hóa kết quả bằng biểu đồ Heatmap, Biplot.

Mỗi dự án được trình bày một cách khoa học và dễ hiểu gồm đề bài yêu cầu, quy trình tổng quát (pipeline) để triển khai dự án, thuật toán tổng quát, các bước tính toán chi tiết, mã lệnh python, trực quan hóa kết quả bằng biểu đồ minh họa.

1 Xây dựng hệ thống AI tìm kiếm tài liệu tương đồng

1.1. Đề bài mô tả yêu cầu

Xét một corpus gồm 06 tài liệu văn bản tiếng Anh:

- D_1: *computer science and information technology*
- D_2: *artificial intelligence and machine learning*
- D_3: *travel to da nang city*
- D_4: *beautiful beaches in my khe*
- D_5: *deep learning for computer vision*
- D_6: *tourism development in vietnam*

Giả sử câu truy vấn (Query) là chính tài liệu:

$$Q = D_1 = \text{“}\textit{computer science and information technology}\text{”}.$$

Mục tiêu:

1. Biểu diễn toàn bộ corpus dưới dạng ma trận TF–IDF (mỗi tài liệu là một vector số thực).
2. Tính độ tương đồng Cosine giữa D_1 và các tài liệu còn lại.
3. Xếp hạng các tài liệu theo mức độ liên quan với truy vấn D_1.
4. Trực quan hoá các vector trong không gian vector (minh họa góc giữa các tài liệu).

1.2. Quy trình tổng quát

Quy trình xử lý có thể tóm tắt như sau:

Tài liệu đầu vào $\rightarrow$ Tiền xử lý $\rightarrow$ Xây dựng từ vựng $\rightarrow$ Tính TF $\rightarrow$ Tính DF, IDF $\rightarrow$

$\rightarrow$ Tính TF–IDF $\rightarrow$ Chuẩn hoá vector $\rightarrow$ Cosine Similarity $\rightarrow$ Xếp hạng.

Bước 1: Tiền xử lý và xây dựng từ vựng

1. Tiền xử lý văn bản

Các bước tiền xử lý thường dùng:

- Chuyển về chữ thường (lowercasing): Ví dụ "Computer" $\rightarrow$ "computer".
- Tokenization (tách từ):

$$D_1 : \text{"computer science and information technology"}$$

$$\Rightarrow [\text{computer, science, and, information, technology}].$$

- Loại bỏ stopwords (các từ ít mang nghĩa nội dung): Các từ như *and, for, in, to, my, ...* thường bị loại bỏ.

Sau khi tiền xử lý với `stop_words='english'` (trong `TfidfVectorizer`), mỗi câu chỉ còn các từ khóa (term) có ý nghĩa.

2. Xây dựng tập từ vựng V

Gộp tất cả từ khóa (sau tiền xử lý) của 6 tài liệu lại và loại các từ trùng, ta thu được tập từ vựng:

$$V = \{t_1, t_2, \ldots, t_{|V|}\}.$$

Ví dụ (một phần):

$$V \approx \left\{ \begin{array}{l} \text{computer, science, information, technology, artificial, intelligence,} \\ \text{machine, learning, travel, da, nang, city, beautiful, beaches, khe,} \\ \text{deep, vision, tourism, development, vietnam} \end{array} \right\}.$$

Mỗi tài liệu D_i sau đó sẽ được biểu diễn dưới dạng một vector trong không gian $\mathbb{R}^{|V|}$.

Bước 2: Tính TF, DF, IDF và TF–IDF

1. Định nghĩa TF (Term Frequency)

Với mỗi từ t_j trong tài liệu D_i, ta định nghĩa:

$$TF_{i,j} = \frac{f_{i,j}}{\sum_k f_{i,k}},$$

trong đó $f_{i,j}$ là số lần xuất hiện của từ t_j trong D_i, và $\sum_k f_{i,k}$ là tổng số từ (sau tiền xử lý) trong D_i.

Ví dụ chi tiết với D_1.

Giả sử sau khi loại bỏ stopwords, D_1 còn:

$$D_1 : [\text{computer, science, information, technology}]$$

Tổng số từ trong D_1 (sau xử lý) là:

$$\sum_k f_{1,k} = 4.$$

Khi đó:

$$TF(\text{computer}, D_1) = \frac{1}{4}, \quad TF(\text{science}, D_1) = \frac{1}{4},$$
$$TF(\text{information}, D_1) = \frac{1}{4}, \quad TF(\text{technology}, D_1) = \frac{1}{4}.$$

2. Định nghĩa DF (Document Frequency)

Với một từ t_j, ta định nghĩa:

$$df(t_j) = \text{số lượng tài liệu trong corpus có chứa } t_j.$$

Ví dụ:

- Từ “computer” xuất hiện trong D_1 và D_5:

$$df(\text{computer}) = 2.$$

- Từ “science” chỉ xuất hiện trong D_1:

$$df(\text{science}) = 1.$$

- Từ "learning" xuất hiện trong D_2 và D_5:

$$df(\text{learning}) = 2.$$

3. Định nghĩa IDF (Inverse Document Frequency)

Để giảm trọng số của những từ xuất hiện ở quá nhiều tài liệu (ít mang tính phân biệt), ta dùng IDF:

$$IDF(t_j) = \log_{10}\left(\frac{N}{df(t_j)}\right),$$

với N là tổng số tài liệu trong corpus. Trong bài toán:

$$N = 6.$$

Ví dụ 1: IDF cho từ "computer"
Từ "computer" xuất hiện trong D_1 và D_5, nên:

$$df(\text{computer}) = 2.$$

Khi đó:

$$IDF(\text{computer}) = \log_{10}\left(\frac{6}{2}\right) = \log_{10}(3).$$

Với log cơ số 10:

$$\log_{10}(3) \approx 0.4771.$$

Ví dụ 2: IDF cho từ "science"

Từ "science" chỉ xuất hiện trong D_1, nên:

$$df(\text{science}) = 1.$$

Khi đó:

$$IDF(\text{science}) = \log_{10}\left(\frac{6}{1}\right) = \log_{10}(6).$$

Với:

$$\log_{10}(6) \approx 0.7782.$$

4. Trọng số TF–IDF

Trọng số TF–IDF của từ t_j trong tài liệu D_i:

$$w_{i,j} = TF_{i,j} \cdot IDF(t_j).$$

Ví dụ 1: TF–IDF của "computer" trong D_1.

Như trên, giả sử $TF(\text{computer}, D_1) = \dfrac{1}{4} = 0.25$, và

$$IDF(\text{computer}) \approx 0.4771.$$

Vậy:

$$w(\text{computer}, D_1) = 0.25 \times 0.4771 \approx 0.1193.$$

Ví dụ 2: TF–IDF của "science" trong D_1.

Ta có:

$$TF(\text{science}, D_1) = \frac{1}{4} = 0.25,$$
$$IDF(\text{science}) \approx 0.7782.$$

Nên:

$$w(\text{science}, D_1) = 0.25 \times 0.7782 \approx 0.1946.$$

Nhận xét. Ta thấy:

$$w(\text{science}, D_1) > w(\text{computer}, D_1),$$

vì từ "science" chỉ xuất hiện trong một tài liệu (tính phân biệt cao hơn), nên IDF lớn hơn. Đây chính là ý nghĩa cốt lõi của IDF: *từ càng hiếm trong corpus thì quan trọng hơn*.

5. Bảng TF–IDF đơn giản cho D_1 đến D_6 (mẫu)

Xét các từ: {computer, science, learning, vision} và sáu tài liệu $D_1, D_2, D_3, D_4, D_5, D_6$.

Với $N = 6$ (tổng số tài liệu), ta có (dùng $\log_{10}$):

$$\text{idf}(\text{computer}) = \log_{10} \frac{6}{2} = \log_{10} 3,$$
$$\text{idf}(\text{science}) = \log_{10} 6,$$
$$\text{idf}(\text{learning}) = \log_{10} 6,$$
$$\text{idf}(\text{vision}) = \log_{10} 6.$$

Kết quả ma trận TF–IDF (với chuẩn TF là tần suất tương đối, IDF là $\log_{10}(N/df)$) là ma trận thưa, các ô trống biểu thị giá trị 0.

Term (Từ)	D_1	D_2	D_3	D_4	D_5	D_6
Length	4	4	4	4	4	3
artificial	0	0.195	0	0	0	0
beaches	0	0	0	0.195	0	0
beautiful	0	0	0	0.195	0	0

city	0	0	0.195	0	0	0
computer	**0.119**	0	0	0	**0.119**	0
da	0	0	0.195	0	0	0
deep	0	0	0	0	0.195	0
development	0	0	0	0	0	0.259
information	0.195	0	0	0	0	0
intelligence	0	0.195	0	0	0	0
khe	0	0	0	0.195	0	0
learning	0	**0.119**	0	0	**0.119**	0
machine	0	0.195	0	0	0	0
my	0	0	0	0.195	0	0
nang	0	0	0.195	0	0	0
science	0.195	0	0	0	0	0
technology	0.195	0	0	0	0	0
tourism	0	0	0	0	0	0.259
travel	0	0	0.195	0	0	0
vietnam	0	0	0	0	0	0.259
vision	0	0	0	0	0.195	0

Bước 3: Biểu diễn mỗi tài liệu dưới dạng vector

Sau khi tính xong TF–IDF cho mọi từ trong từ vựng V, với mỗi tài liệu D_i ta thu được một vector:

$$\vec{v}_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,|V|}) \in \mathbb{R}^{|V|}.$$

Ví dụ, nếu xét thứ tự từ vựng đầy đủ:

$$V = (\text{computer}, \text{science}, \text{information}, \text{technology}, \ldots),$$

thì vector của D_1 có dạng:

$$\vec{v}_1 = (w(\text{computer}, D_1), w(\text{science}, D_1), w(\text{information}, D_1), w(\text{technology}, D_1), \ldots).$$

Các tài liệu không chứa một từ nào đó thì tại vị trí tương ứng trọng số bằng 0.

Ví dụ cụ thể (với từ vựng rút gọn). Xét từ vựng rút gọn:

$$V' = (\text{computer}, \text{science}, \text{learning}, \text{vision}),$$

và 5 tài liệu $D_1, \ldots, D_5$.

Với $N = 5$ và df như đã tính, ta có:

$$\begin{aligned}\text{idf}(\text{computer}) &= \ln\frac{5}{2},\\ \text{idf}(\text{science}) &= \ln 5,\\ \text{idf}(\text{learning}) &= \ln 5,\\ \text{idf}(\text{vision}) &= \ln 5.\end{aligned}$$

Khi đó, các vector TF–IDF (sắp xếp đúng theo thứ tự của V') là:

$$\begin{aligned}\vec{v}_1 &= \big(w(\text{computer},D_1), w(\text{science},D_1), w(\text{learning},D_1), w(\text{vision},D_1)\big)\\ &= \big(\ln\tfrac{5}{2},\ \ln 5,\ 0,\ 0\big) \approx (0.9163,\ 1.6094,\ 0,\ 0),\\ \vec{v}_2 &= (0,\ 0,\ 0,\ 0),\\ \vec{v}_3 &= (0,\ 0,\ 0,\ 0),\\ \vec{v}_4 &= (0,\ 0,\ 0,\ 0),\\ \vec{v}_5 &= \big(w(\text{computer},D_5), w(\text{science},D_5), w(\text{learning},D_5), w(\text{vision},D_5)\big)\\ &= \big(\ln\tfrac{5}{2},\ 0,\ \ln 5,\ \ln 5\big) \approx (0.9163,\ 0,\ 1.6094,\ 1.6094).\end{aligned}$$

Như vậy, thứ tự thành phần trong vector luôn tuân theo thứ tự từ trong tập từ vựng $(\text{computer}, \text{science}, \text{learning}, \text{vision})$; nếu tài liệu không chứa từ nào đó thì trọng số tại vị trí tương ứng bằng 0.

Chuẩn hoá L2:

Để tránh sự ảnh hưởng của độ dài văn bản, ta chuẩn hóa mỗi vector theo chuẩn L2:

$$\|\vec{v}_i\|_2 = \sqrt{\sum_{j=1}^{|V|} w_{i,j}^2},$$

$$\vec{v}_i' = \frac{\vec{v}_i}{\|\vec{v}_i\|_2}.$$

Sau chuẩn hoá, ta có:

$$\|\vec{v}_i'\|_2 = 1.$$

Việc chuẩn hoá giúp:

- Chỉ còn hướng của vector là quan trọng (nội dung từ vựng).
- Độ dài tuyệt đối (số từ nhiều hay ít) không ảnh hưởng trực tiếp đến độ tương đồng.

Bước 4: Tính Cosine Similarity và giải thích

1. Định nghĩa Cosine Similarity

Độ tương đồng Cosine giữa hai vector (đã chuẩn hoá) $\vec{v}'_1$ và $\vec{v}'_5$ được định nghĩa:

$$\cos(\theta) = \vec{v}'_1 \cdot \vec{v}'_5 = \sum_{j=1}^{|V|} v'_{1,j} v'_{5,j}.$$

Do các vector đã được chuẩn hoá có độ dài bằng 1, nên:

$$\cos(\theta) = \frac{\vec{v}_1 \cdot \vec{v}_5}{\|\vec{v}_1\| \, \|\vec{v}_5\|}.$$

2. Cách tính cụ thể tích vô hướng

Khi tính tích vô hướng:

$$\vec{v}'_1 \cdot \vec{v}'_5 = \sum_{j=1}^{|V|} v'_{1,j} v'_{5,j},$$

chỉ những từ xuất hiện trong cả hai tài liệu mới đóng góp vào giá trị này.

Trong ví dụ này, D_1 và D_5 chỉ có từ chung là "computer".

Giả sử sau chuẩn hoá:

$$v'_{1,\text{computer}} = 0.5, \quad v'_{5,\text{computer}} = 0.5.$$

Các thành phần tương ứng với những từ khác hoặc bằng 0 (không cùng xuất hiện), hoặc không ảnh hưởng đến tích vô hướng.

Do đó:

$$\vec{v}'_1 \cdot \vec{v}'_5 = 0.5 \times 0.5 + (\text{các hạng tử khác} = 0) = 0.25.$$
$$\Rightarrow \cos(\theta_{1,5}) = 0.25.$$

Vì $0 < 0.25 < 1$ nên D_5 có mức độ tương đồng nhất định với D_1 (chung ý nghĩa về "computer").

3. Trường hợp không có từ chung

Xét D_1 và D_3. Giả sử sau khi xử lý, chúng không có từ chung. Khi đó:

$$\forall j, \quad \text{hoặc } v'_{1,j} = 0 \text{ hoặc } v'_{3,j} = 0.$$

Nên:

$$\vec{v}'_1 \cdot \vec{v}'_3 = \sum_j v'_{1,j} v'_{3,j} = 0.$$

$$\Rightarrow \cos(\theta_{1,3}) = 0 \Rightarrow \theta = 90^\circ.$$

Điều này được hiểu là hai vector vuông góc trong không gian vector và hai tài liệu không liên quan về nội dung từ vựng.

4. Ma trận Cosine Similarity

Khi dùng các vector TF–IDF (chuẩn hoá L2) của 6 tài liệu, ta thu được ma trận:

$$\text{sim_matrix} \in \mathbb{R}^{6\times 6},$$

trong đó:

$$\text{sim_matrix}[i,j] = \cos(\theta_{i,j}) = \text{độ tương đồng giữa } D_i \text{ và } D_j.$$

Một số tính chất:

- Ma trận đối xứng: $\text{sim_matrix}[i,j] = \text{sim_matrix}[j,i]$.
- $\text{sim_matrix}[i,i] = 1$ (mỗi tài liệu luôn trùng khớp hoàn toàn với chính nó).

Với 6 tài liệu $(D_1, \ldots, D_6)$ và các vector TF–IDF đã chuẩn hoá, ma trận cosine similarity thu được (làm tròn 4 chữ số thập phân) là:

$$\text{sim_matrix} \approx \begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1233 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.1233 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ 0.1233 & 0.1233 & 0.0000 & 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}.$$

Hàng thứ 1 của ma trận (tương ứng với D_1) là:

$$\begin{aligned} &\big[\cos(\theta_{1,1}), \cos(\theta_{1,2}), \cos(\theta_{1,3}), \cos(\theta_{1,4}), \cos(\theta_{1,5}), \cos(\theta_{1,6})\big] \\ &= [1.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.1233,\ 0.0000]. \end{aligned}$$

Bước 5: Xếp hạng và lựa chọn Top-k

Trong bước này, ta tiến hành xếp hạng mức độ liên quan giữa tài liệu truy vấn D_1 và các tài liệu còn lại dựa trên độ tương đồng Cosine. Giả sử ta chọn $k = 3$ (Top-3 tài liệu giống nhất với D_1).

1. Lấy hàng tương ứng với D_1 trong ma trận Cosine Similarity

Từ ma trận cosine similarity đã tính ở bước trước, hàng thứ nhất (tương ứng với tài liệu D_1) có dạng:

$$\text{scores} = [1.0000,\ 0.0000,\ 0.0000,\ 0.0000,\ 0.1233,\ 0.0000]\,.$$

Ý nghĩa từng phần tử:

$$\begin{aligned}
&\cos(\theta_{1,1}) = 1.0000 && \text{(trùng khớp với chính nó)},\\
&\cos(\theta_{1,2}) = 0 && \text{(không có từ chung với } D_2\text{)},\\
&\cos(\theta_{1,3}) = 0 && \text{(không có từ chung với } D_3\text{)},\\
&\cos(\theta_{1,4}) = 0 && \text{(không có từ chung với } D_4\text{)},\\
&\cos(\theta_{1,5}) = 0.1233 && \text{(có chung từ “computer” với } D_5\text{)},\\
&\cos(\theta_{1,6}) = 0 && \text{(không có từ chung với } D_6\text{)}.
\end{aligned}$$

2. Loại bỏ chính tài liệu truy vấn

Do không xếp hạng tài liệu đối với chính nó, ta loại bỏ phần tử đầu tiên:

$$\text{scores}' = [0,\ 0,\ 0,\ 0.1233,\ 0].$$

Vector này tương ứng với các tài liệu:

$$(D_2,\ D_3,\ D_4,\ D_5,\ D_6).$$

3. Sắp xếp thứ tự giảm dần độ tương đồng

Ta sắp xếp scores$'$ theo thứ tự giảm dần:

$$0.1233 > 0 = 0 = 0 = 0.$$

Do đó, thứ tự ưu tiên là:

$$D_5,\ D_2,\ D_3,\ D_4,\ D_6.$$

4. Lựa chọn Top-k

Với $k = 3$, ta chọn ba tài liệu có giá trị cosine cao nhất:

$$\boxed{\{D_5,\ D_2,\ D_3\}}.$$

Trong đó:

$$\cos(\theta_{1,5}) \approx 0.1233, \qquad \cos(\theta_{1,2}) = \cos(\theta_{1,3}) = \cos(\theta_{1,4}) = \cos(\theta_{1,6}) = 0.$$

Các tài liệu D_2, D_3, D_4, D_6 cùng có độ tương đồng bằng 0, do đó việc chọn D_2 và D_3 hoàn toàn tương đương với chọn bất kỳ hai tài liệu trong nhóm $\{D_2, D_3, D_4, D_6\}$.

5. Giải thích tại sao D_5 là tài liệu gần nhất với D_1

Từ bảng TF–IDF ở mục trên, với việc chỉ xét bốn thành phần $\{\text{computer}, \text{science}, \text{learning}, \text{vision}\}$, các trọng số (chưa chuẩn hoá) của D_1 và D_5 xấp xỉ:

$$\begin{aligned}\vec{v}_1 &= \big(w(\text{computer}, D_1), w(\text{science}, D_1), w(\text{learning}, D_1), w(\text{vision}, D_1)\big)\\ &\approx (0.1193,\ 0.1946,\ 0,\ 0),\\ \vec{v}_5 &= \big(w(\text{computer}, D_5), w(\text{science}, D_5), w(\text{learning}, D_5), w(\text{vision}, D_5)\big)\\ &\approx (0.1193,\ 0,\ 0.1193,\ 0.1946).\end{aligned}$$

Trong toàn bộ không gian đặc trưng, hai tài liệu D_1 và D_5 chỉ cùng chia sẻ từ khóa `computer`, nên tích vô hướng giữa hai vector TF–IDF (chưa chuẩn hoá) là:

$$\vec{v}_1 \cdot \vec{v}_5 = w(\text{computer}, D_1)\, w(\text{computer}, D_5) \approx 0.1193^2 \approx 0.0142.$$

Chuẩn L2 của hai vector đầy đủ lần lượt là:

$$\|\vec{v}_1\|_2 \approx 0.3574, \qquad \|\vec{v}_5\|_2 \approx 0.3227.$$

Do đó, độ tương đồng Cosine giữa D_1 và D_5 đúng bằng:

$$\cos(\theta_{1,5}) = \frac{\vec{v}_1 \cdot \vec{v}_5}{\|\vec{v}_1\|_2 \, \|\vec{v}_5\|_2} \approx \frac{0.0142}{0.3574 \times 0.3227} \approx 0.1233,$$

hoàn toàn trùng với giá trị trong ma trận ở mục trên.

Các tài liệu D_2, D_3, D_4, D_6 không chia sẻ bất kỳ từ khóa nội dung nào với D_1 nên tích vô hướng bằng 0, dẫn tới độ tương đồng Cosine bằng 0.

6. Kết luận bước xếp hạng

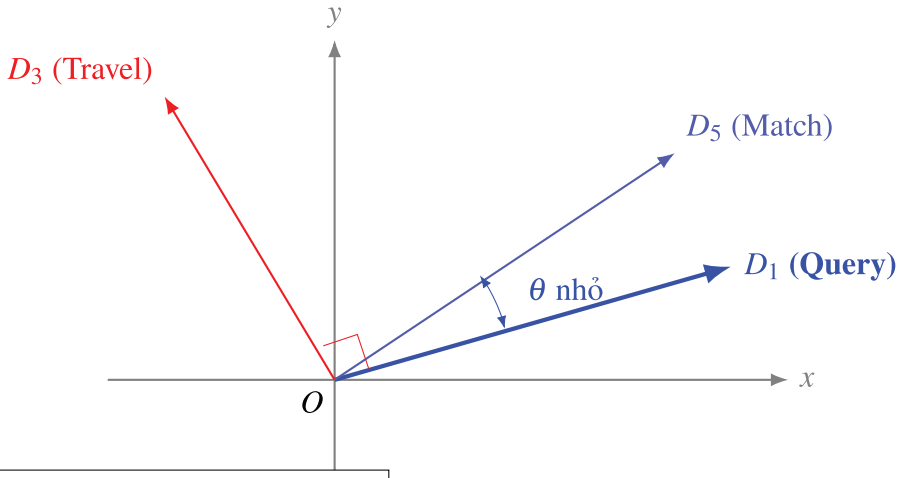
D_5 là tài liệu liên quan nhất (độ tương đồng cao nhất) với D_1.

1.3. Trực quan hoá trong không gian vector

Để trực quan hoá kết quả, ta vẽ các vector đại diện cho tài liệu trong mặt phẳng 2D (chỉ mô phỏng, không phải toạ độ thực sự của TF–IDF).

Ý tưởng:

- D_1 và D_5 tạo thành một góc nhọn (cosine > 0).
- D_1 và D_3 gần vuông góc (cosine ≈ 0).



Chú giải:
- D_1, D_5: Chung từ "computer" $\Rightarrow \cos(\theta) > 0$.
- D_1, D_3: Không có từ chung $\Rightarrow \cos(\theta) \approx 0$.
- $\cos(\theta)$ càng lớn $\Rightarrow$ tài liệu càng tương đồng.

Hình 1: Minh họa Cosine Similarity giữa các tài liệu trong không gian vector.

1.4. Kết luận và mở rộng bài toán

Trong bài toán này, chúng ta đã:

- Biểu diễn tài liệu văn bản dưới dạng vector số thực bằng TF–IDF.
- Hiểu rõ từng bước: TF, DF, IDF, TF–IDF, chuẩn hoá L2.
- Tính độ tương đồng Cosine giữa truy vấn và các tài liệu.
- Xếp hạng tài liệu theo mức độ tương đồng.
- Trực quan hóa ý nghĩa của Cosine Similarity bằng góc giữa các vector.

Hướng mở rộng:

- Thay TF–IDF bằng các mô hình nâng cao (BM25, Word2Vec, Doc2Vec).
- Áp dụng cho corpus lớn với hàng nghìn hoặc hàng triệu tài liệu.
- Kết hợp với các kỹ thuật truy vấn ngôn ngữ tự nhiên nâng cao.

1.5. Mã Python (kèm chú thích chi tiết)

Bạn đọc có thể quét QR Code để truy cập và chạy chương trình tại Google Colab:

```
# ==========================================
# SETUP ENVIRONMENT & LIBRARIES
# ==========================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import
   TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Configure display options for better readability
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 1000)
pd.set_option('display.float_format', '{:.4f}'.format)

# ==========================================
# PART 1: INPUT DATA & PREPROCESSING
# ==========================================
print("--- PART 1: INPUT DATA ---")

# 1. Define Corpus (6 documents)
documents = {
    "D1": "computer science and information technology",
    "D2": "artificial intelligence and machine learning",
    "D3": "travel to da nang city",
    "D4": "beautiful beaches in my khe",
    "D5": "deep learning for computer vision",
    "D6": "tourism development in vietnam"
}

# Convert dictionary to lists
corpus_ids = list(documents.keys())
corpus_content = list(documents.values())

# Create DataFrame to display raw data
df_raw = pd.DataFrame(list(documents.items()), columns=['
   ID', 'Content (Raw Text)'])
print(df_raw)
print("\n")
```

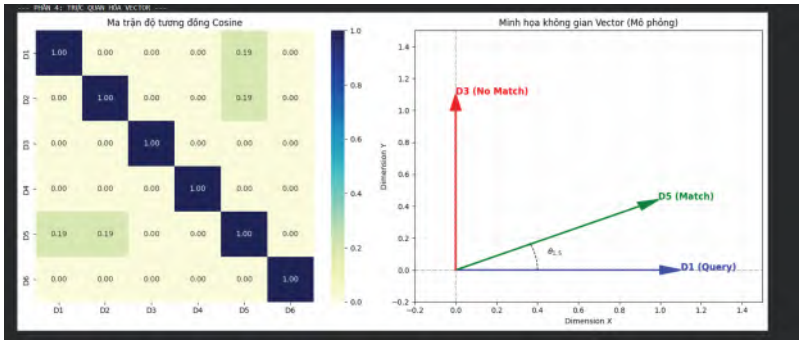
```
# ==========================================
# PART 2: TF-IDF CALCULATION
# ==========================================
print("--- PART 2: TF-IDF MATRIX ---")

# Initialize TfidfVectorizer
# stop_words='english': Remove common words (and, in, for
    , to...)
vectorizer = TfidfVectorizer(stop_words='english')

# Fit and transform the corpus
tfidf_matrix = vectorizer.fit_transform(corpus_content)

# Get feature names (vocabulary)
feature_names = vectorizer.get_feature_names_out()

# Convert to DataFrame for visualization
df_tfidf = pd.DataFrame(tfidf_matrix.toarray(), index=
    corpus_ids, columns=feature_names)

print("TF-IDF Matrix (Stopwords removed):")
# Display only columns relevant to D1 and D5 for clarity
sample_cols = [col for col in feature_names if df_tfidf.
    loc['D1', col] > 0 or df_tfidf.loc['D5', col] > 0]
print(df_tfidf[sample_cols].T)
print(f"\nFull matrix shape: {df_tfidf.shape}")
print("\n")

# ==========================================
# PART 3: COSINE SIMILARITY & RANKING
# ==========================================
print("--- PART 3: SIMILARITY & RANKING ---")

# Compute Cosine Similarity Matrix
cosine_sim_matrix = cosine_similarity(tfidf_matrix,
    tfidf_matrix)

# Create DataFrame
df_sim = pd.DataFrame(cosine_sim_matrix, index=corpus_ids
    , columns=corpus_ids)

print("Cosine Similarity Matrix:")
print(df_sim)
print("\n")
```

```
# --- RANKING FOR QUERY D1 ---
query_id = "D1"
print(f"Search Results for Query: {query_id} ('{documents
    [query_id]}')")

# Get scores for D1
d1_scores = df_sim.loc[query_id]

# Sort descending
ranking = d1_scores.sort_values(ascending=False)

# Drop the query itself from results
ranking = ranking.drop(query_id)

# Print ranked results
for idx, (doc_id, score) in enumerate(ranking.items(), 1)
    :
     print(f"Rank {idx}: {doc_id} - Score: {score:.4f} ->
    Content: {documents[doc_id]}")

print("\n=> CONCLUSION: D5 is the most relevant document
    (shares 'computer').")
print("\n")

# ==========================================
# PART 4: VISUALIZATION (VECTOR SPACE)
# ==========================================
print("--- PART 4: VISUALIZATION ---")

plt.figure(figsize=(14, 6))

# --- PLOT 1: SIMILARITY HEATMAP ---
plt.subplot(1, 2, 1)
sns.heatmap(df_sim, annot=True, cmap='Blues', fmt='.2f',
    cbar=True)
plt.title('Cosine Similarity Matrix')
plt.xlabel('Document IDs')
plt.ylabel('Document IDs')

# --- PLOT 2: VECTOR SPACE ILLUSTRATION (2D Simulation)
    ---
plt.subplot(1, 2, 2)

# Helper function to plot vectors
def plot_vector(vec, label, color):
    # Draw arrow
```

```
    plt.arrow(0, 0, vec[0], vec[1], head_width=0.05,
   head_length=0.1,
              fc=color, ec=color, linewidth=2,
   length_includes_head=True)
    # Add label
    plt.text(vec[0]*1.15, vec[1]*1.15, label, color=color
   ,
             fontsize=11, fontweight='bold', ha='center')

# Simulate 2D coordinates for illustration purposes
# D1 is the baseline (x-axis)
vec_d1 = np.array([1.0, 0.0])
# D5 has similarity ~0.12 (small positive angle, but
   simplified here as visible acute angle)
vec_d5 = np.array([0.9, 0.4])
# D3 has similarity 0.0 (orthogonal/90 degrees)
vec_d3 = np.array([0.0, 1.0])

# Draw axes
plt.axhline(y=0, color='gray', linestyle='--', alpha=0.5)
plt.axvline(x=0, color='gray', linestyle='--', alpha=0.5)

# Plot vectors
plot_vector(vec_d1, "D1 (Query)", 'blue')
plot_vector(vec_d5, "D5 (Match)", 'green')
plot_vector(vec_d3, "D3 (No Match)", 'red')

# Draw angle arc for D1 and D5
theta = np.linspace(0, 24, 100) # Arc from 0 to ~24
   degrees
x_arc = 0.5 * np.cos(np.radians(theta))
y_arc = 0.5 * np.sin(np.radians(theta))
plt.plot(x_arc, y_arc, 'k:', linewidth=1.5)
plt.text(0.55, 0.15, r'Small $\theta$', fontsize=10)

# Draw right angle symbol for D3
plt.plot([0, 0.1, 0.1], [0.1, 0.1, 0], 'r-', linewidth=1)

# Plot settings
plt.xlim(-0.2, 1.5)
plt.ylim(-0.2, 1.3)
plt.title('Vector Space Illustration (2D Projection)')
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.grid(True, alpha=0.3)
```

```
# Add Legend box
textstr = '\n'.join((
    r'$D_1 \cdot D_5 > 0$: Similar',
    r'$D_1 \cdot D_3 = 0$: Unrelated',
    r'Cosine = $\cos(\theta)$'
    ))
props = dict(boxstyle='round', facecolor='white', alpha
   =0.8)
plt.text(0.8, 0.9, textstr, transform=plt.gca().transAxes
   , fontsize=10,
        verticalalignment='top', bbox=props)

plt.tight_layout()
plt.show()
```

Kết quả chương trình:



Hình 2: Kết quả chạy chương trình

Giải thích mã nguồn:

- `TfidfVectorizer` tự động:
  - Tách từ.
  - Loại bỏ stopwords.
  - Tính TF.
  - Tính IDF trên toàn bộ corpus.
  - Nhân TF $\times$ IDF để tạo ma trận TF–IDF.
- `cosine_similarity(X_tfidf)`:
  - Tính Cosine Similarity cho tất cả các cặp hàng (tài liệu).
  - Trả về ma trận 6×6.

- `np.argsort(scores)[::-1]`:
  - Sắp xếp chỉ số tài liệu theo thứ tự điểm giảm dần.
  - Cho phép ta xếp hạng tài liệu theo độ liên quan.

2 Xây dựng hệ thống AI phân cụm văn bản

2.1. Đề bài mô tả yêu cầu

Xét cùng corpus 6 tài liệu tiếng Anh như Bài 1:

- D_1: *computer science and information technology*
- D_2: *artificial intelligence and machine learning*
- D_3: *travel to da nang city*
- D_4: *beautiful beaches in my khe*
- D_5: *deep learning for computer vision*
- D_6: *tourism development in vietnam*

Mục tiêu:

1. Biểu diễn từng tài liệu bằng vector Word2Vec (semantic embedding).
2. Giảm chiều bằng SVD/Truncated SVD xuống không gian 2D.
3. Phân cụm 6 tài liệu bằng K-Means.
4. Trực quan hoá phân cụm trong không gian 2D.

2.2. Quy trình tổng quát

Quy trình xử lý với Word2Vec được mô tả như sau:

$$\text{Tài liệu đầu vào} \to \text{Tiền xử lý} \to \text{Huấn luyện Word2Vec} \to \text{Embedding tài liệu} \to \text{Giảm chiều bằng SVD} \to \text{K-Means} \to \text{Trực quan hoá}.$$

Mục tiêu cuối cùng: khám phá *cấu trúc ngữ nghĩa* và *nhóm chủ đề* trong dữ liệu văn bản.

Bước 1: Tiền xử lý văn bản

Chúng ta áp dụng tiền xử lý như Bài 1:

- chuyển về chữ thường,

- tách từ (tokenization),
- loại bỏ dấu câu,
- loại bỏ stopwords.

Ví dụ:

$$D_1 \Rightarrow [\text{computer, science, information, technology}].$$

Các tài liệu sau tiền xử lý được dùng để huấn luyện Word2Vec.

Bước 2: Word2Vec – Biểu diễn từ dưới dạng vector

1. Ý tưởng

Word2Vec ánh xạ mỗi từ sang một vector trong $\mathbb{R}^d$:

$$\vec{e}(t) \in \mathbb{R}^d, \quad d \in \{50, 100, 300\}.$$

Hai mô hình:

- CBOW – dự đoán từ dựa trên ngữ cảnh.
- Skip-gram – dự đoán ngữ cảnh dựa trên từ trung tâm.

Trong bài này, ta dùng Skip-gram (thường cho kết quả tốt hơn với corpus nhỏ).

Word2Vec là một mô hình học biểu diễn phân bố (distributed representation) của từ vựng. Sau quá trình huấn luyện, mô hình tạo ra hai thành phần chính:

1. Từ điển (Vocabulary): danh sách tất cả các từ xuất hiện đủ nhiều trong corpus.
2. Ma trận embedding $\mathbf{E}$: một ma trận kích thước $|V| \times d$, trong đó:

$$\mathbf{E} = \begin{bmatrix} - & \vec{e}(t_1) & - \\ - & \vec{e}(t_2) & - \\ & \vdots & \\ - & \vec{e}(t_{|V|}) & - \end{bmatrix}$$

   mỗi hàng chính là vector biểu diễn của một từ.

1) Từ điển sau huấn luyện.

Ví dụ với corpus:

$$D_1, \ldots, D_6,$$

sau tiền xử lý, Word2Vec sẽ xây dựng từ điển:

$$V = \{\text{computer, science, learning, travel, beaches, vision, tourism, ...}\}.$$

Từ điển quyết định:

- từ nào sẽ có vector,
- từ nào bị loại (vì quá hiếm theo `min_count`).

2) Ma trận embedding sau huấn luyện.

Word2Vec gán cho mỗi từ t một vector chiều d (ví dụ $d = 100$):

$$\vec{e}(t) = (e_{t,1}, e_{t,2}, \ldots, e_{t,d}).$$

Các giá trị này được học bằng tối ưu hoá hàm mất mát (CBOW hoặc Skip-gram) (sẽ được chi tiết sau). Điểm quan trọng là các giá trị embedding không được lập trình sẵn, mà được học từ dữ liệu.

Ví dụ: mô hình học được (minh hoạ).

Giả sử sau huấn luyện, mô hình thu được:

$$\begin{aligned}
\vec{e}(\text{computer}) &= (0.20,\ 0.10,\ -0.05,\ 0.30, \ldots),\\
\vec{e}(\text{science}) &= (0.18,\ 0.05,\ -0.02,\ 0.25, \ldots),\\
\vec{e}(\text{travel}) &= (-0.10,\ 0.40,\ 0.22,\ -0.05, \ldots),\\
\vec{e}(\text{beaches}) &= (-0.15,\ 0.35,\ 0.30,\ -0.08, \ldots).
\end{aligned}$$

Các từ cùng chủ đề (ví dụ computer, science) sẽ có vector nằm gần nhau. Ngược lại, các từ khác chủ đề (ví dụ travel, beaches) sẽ xa vùng của "AI".

Lý do là vì trong quá trình huấn luyện:

- Skip-gram: từ nào thường xuất hiện cùng nhau $\rightarrow$ vector gần nhau.
- CBOW: từ nào có chung ngữ cảnh $\rightarrow$ vector gần nhau.

Do đó, Word2Vec không chỉ biểu diễn sự xuất hiện của từ, mà còn biểu diễn được mối quan hệ ngữ nghĩa.

Tóm lại:

Sau khi huấn luyện Word2Vec, ta thu được:

- một từ điển V,
- mỗi từ được biểu diễn bởi một vector chiều d,
- các vector nằm trong không gian ngữ nghĩa chung,
- vector gần nhau $\rightarrow$ từ có ý nghĩa gần nhau.

Nhờ đó, ta mới có thể tính vector tài liệu bằng trung bình embedding các từ.

2. Vector tài liệu từ Word2Vec

Sau khi huấn luyện Word2Vec, mỗi từ t trong từ điển đều có một vector ngữ nghĩa $\vec{e}(t) \in \mathbb{R}^d$. Tuy nhiên, nhiệm vụ của chúng ta là biểu diễn mỗi tài liệu (document) dưới dạng một vector duy nhất để có thể thực hiện phân cụm hay phân loại.

Vì một tài liệu chứa nhiều từ, cách đơn giản và hiệu quả nhất là lấy trung bình các vector từ trong tài liệu:

$$\vec{v}_i = \frac{1}{|D_i|} \sum_{t_j \in D_i} \vec{e}(t_j).$$

Trong đó:

- D_i: tài liệu thứ i,
- $|D_i|$: số lượng token (từ) trong tài liệu sau tiền xử lý,
- $\vec{e}(t_j)$: vector embedding của từ thứ j trong tài liệu.

Ý nghĩa trực quan.

Mỗi từ là một điểm trong không gian d chiều. Một tài liệu là trung bình (centroid) của các điểm đó, hay nói cách khác: Vector tài liệu $\vec{v}_i$ là *trọng tâm ngữ nghĩa* của các từ trong tài liệu.

Nhờ đó:

- tài liệu chứa nhiều từ cùng chủ đề sẽ có vector nằm gần các từ đó,
- các tài liệu có chủ đề tương tự nhau sẽ có vector ở gần nhau.

Ưu điểm so với TF–IDF.

- Word2Vec không chỉ xem xét tần suất từ, mà còn mô hình hoá mối quan hệ ngữ nghĩa giữa các từ.
- Các từ như *computer*, *science*, *technology* có embedding gần nhau $\rightarrow$ vector tài liệu cũng nằm gần cụm AI/CS.
- Phù hợp cho phân cụm, vì không gian embedding Word2Vec đã có cấu trúc ngữ nghĩa.

Ví dụ minh hoạ chi tiết.

Giả sử Word2Vec học embedding với $d = 4$ (để minh hoạ dễ hiểu). Trong thực tế $d = 100$ hoặc 300.

Các vector từ (giả lập):

$$\begin{aligned}\vec{e}(\text{computer}) &= (0.20,\ 0.10,\ -0.05,\ 0.30),\\ \vec{e}(\text{science}) &= (0.18,\ 0.05,\ -0.02,\ 0.25),\\ \vec{e}(\text{information}) &= (0.22,\ 0.12,\ -0.01,\ 0.28),\\ \vec{e}(\text{technology}) &= (0.19,\ 0.08,\ 0.00,\ 0.32).\end{aligned}$$

Như vậy tài liệu:

$$D_1 = [\text{computer, science, information, technology}]$$

có vector embedding được tính như sau:

$$\vec{v}_1 = \frac{\vec{e}(\text{computer}) + \vec{e}(\text{science}) + \vec{e}(\text{information}) + \vec{e}(\text{technology})}{4}.$$

Tính toán từng chiều một cách tường minh.

Cộng từng chiều:

$$\begin{aligned}&\text{Chiều 1: } 0.20 + 0.18 + 0.22 + 0.19 = 0.79,\\ &\text{Chiều 2: } 0.10 + 0.05 + 0.12 + 0.08 = 0.35,\\ &\text{Chiều 3: } -0.05 - 0.02 - 0.01 + 0.00 = -0.08,\\ &\text{Chiều 4: } 0.30 + 0.25 + 0.28 + 0.32 = 1.15.\end{aligned}$$

Lấy trung bình:

$$\vec{v}_1 = \left(\frac{0.79}{4},\ \frac{0.35}{4},\ \frac{-0.08}{4},\ \frac{1.15}{4}\right) = (0.1975,\ 0.0875,\ -0.0200,\ 0.2875).$$

Kết luận.

Vector biểu diễn tài liệu D_1 là:

$$\vec{v}_1 \approx (0.1975,\ 0.0875,\ -0.0200,\ 0.2875).$$

Mặc dù ví dụ chỉ dùng $d = 4$, nhưng trong thực tế mỗi tài liệu được biểu diễn bằng vector kích thước $d = 100$ hoặc $d = 300$, mang theo rất nhiều thông tin ngữ nghĩa.

Để hình dung trực quan, ta giảm chiều không gian về 2D:

- các từ *computer, science, technology* nằm gần nhau,
- trung bình của chúng cũng nằm trong cùng vùng $\rightarrow$ tài liệu D_1 nằm trong "cụm AI",
- các từ về du lịch (*travel, beach, tourism*) sẽ nằm ở cụm khác.

Bước 3: Giảm chiều bằng SVD

Sau khi biểu diễn mỗi tài liệu D_i bằng một vector Word2Vec $\vec{v}_i \in \mathbb{R}^d$, ta thu được ma trận đặc trưng:

$$X \in \mathbb{R}^{6\times d}$$

trong đó:

- mỗi hàng của X là một tài liệu (6 tài liệu $\Rightarrow$ 6 hàng),
- mỗi cột là một chiều (feature) trong không gian embedding Word2Vec (thường $d = 100$ hoặc 300).

Không gian có số chiều lớn (d lớn) gây khó khăn:

- khó trực quan hoá (không thể vẽ hình ở $\mathbb{R}^{100}$),
- thuật toán phân cụm (K-Means) có thể kém ổn định,
- tốn tài nguyên tính toán khi số tài liệu tăng.

Do đó, ta sử dụng Truncated SVD (còn gọi là Latent Semantic Analysis (LSA) khi áp dụng trong xử lý ngôn ngữ) để giảm số chiều xuống k (ví dụ $k = 2$ hoặc $k = 3$).

1. Nhắc lại SVD

Với một ma trận $X \in \mathbb{R}^{m\times n}$, phân rã SVD đầy đủ có dạng:

$$X = U\Sigma V^\top$$

trong đó:

- $U \in \mathbb{R}^{m\times m}$: ma trận trực chuẩn (các cột trực giao),
- $V \in \mathbb{R}^{n\times n}$: ma trận trực chuẩn,
- $\Sigma \in \mathbb{R}^{m\times n}$: ma trận đường chéo (các phần tử không âm $\sigma_1 \geq \sigma_2 \geq \dots$ gọi là *singular values*).

Trong bài toán này:

$$X \in \mathbb{R}^{6\times d}, \quad m = 6, \quad n = d.$$

Các giá trị suy biến (singular values) trong Σ thể hiện:

- mức độ quan trọng (độ biến thiên) của từng thành phần ẩn,
- σ_1 càng lớn thì chiều tương ứng càng mang nhiều thông tin.

2. Truncated SVD (SVD rút gọn)

Thay vì giữ toàn bộ các thành phần, ta chỉ giữ lại k thành phần quan trọng nhất (ứng với k singular values lớn nhất). Ta viết xấp xỉ:

$$X \approx U_k \Sigma_k V_k^\top$$

với:

- $U_k \in \mathbb{R}^{6\times k}$,
- $\Sigma_k \in \mathbb{R}^{k\times k}$,
- $V_k \in \mathbb{R}^{d\times k}$.

Khi đó, ta lấy:

$$X_k = U_k \Sigma_k \in \mathbb{R}^{6\times k}$$

làm ma trận đặc trưng giảm chiều, trong đó:

- mỗi hàng của X_k là một vector mới biểu diễn tài liệu,
- số chiều chỉ còn k thay vì d (với $k \ll d$).

Trong bài toán này, nếu ta chọn $k = 2$:

$$X \in \mathbb{R}^{6\times d} \quad \Rightarrow \quad X_2 = U_2 \Sigma_2 \in \mathbb{R}^{6\times 2}.$$

3. Ý nghĩa của việc giảm chiều

Việc sử dụng Truncated SVD mang lại nhiều lợi ích:

- Giữ lại thông tin quan trọng nhất: SVD sắp xếp các thành phần theo mức độ quan trọng (qua singular values), nên việc giữ lại k thành phần lớn nhất tương đương với việc *giữ lại phần lớn năng lượng/độ biến thiên của dữ liệu*.
- Loại bỏ nhiễu (noise): Các thành phần có singular value rất nhỏ thường gắn với nhiễu, thông tin ít hữu ích cho phân cụm $\rightarrow$ loại bỏ chúng giúp K-Means hoạt động ổn định hơn.
- Thuận tiện cho trực quan hoá: Khi $k = 2$ hoặc $k = 3$, ta có thể vẽ được các điểm (tài liệu) trên mặt phẳng 2D hoặc không gian 3D, dễ dàng quan sát các cụm chủ đề.
- Giảm chi phí tính toán: Thay vì làm việc trong không gian d chiều (100–300 chiều), ta chỉ cần làm việc trong không gian k chiều nhỏ hơn rất nhiều.

4. Mô tả một cách trực quan

Có thể hình dung:

- Ban đầu, mỗi tài liệu là một điểm trong không gian 100 chiều (embedding Word2Vec).

- SVD tìm ra các trục ẩn (latent axes) mà dọc theo đó dữ liệu thay đổi mạnh nhất.
- Khi ta chiếu dữ liệu lên 2 trục quan trọng nhất, các tài liệu có nội dung giống nhau sẽ nằm gần nhau trên mặt phẳng 2D.

Sau bước này, ta thu được:

$$X_2 = \begin{bmatrix} - & \vec{z}_1^\top & - \\ - & \vec{z}_2^\top & - \\ & \vdots & \\ - & \vec{z}_6^\top & - \end{bmatrix} \in \mathbb{R}^{6\times 2},$$

trong đó $\vec{z}_i \in \mathbb{R}^2$ là vector giảm chiều của tài liệu D_i, được dùng làm đầu vào cho bước tiếp theo: phân cụm bằng K-Means.

Bước 4: Phân cụm bằng K-Means

Sau khi huấn luyện Word2Vec và tính được vector embedding cho từng tài liệu (Bước 2), ta thu được ma trận đặc trưng:

$$X = \begin{bmatrix} - & \vec{v}_1^\top & - \\ - & \vec{v}_2^\top & - \\ & \vdots & \\ - & \vec{v}_6^\top & - \end{bmatrix} \in \mathbb{R}^{6\times d},$$

trong đó d là số chiều embedding (ví dụ $d = 100$).

Để trực quan hoá và làm thuật toán phân cụm ổn định hơn, ta giảm chiều dữ liệu bằng SVD xuống $k = 2$ như Bước 3:

$$X_2 = U_2\Sigma_2 \in \mathbb{R}^{6\times 2}.$$

Mỗi tài liệu D_i được biểu diễn bởi vector 2 chiều:

$$\vec{z}_i \in \mathbb{R}^2.$$

1. Mục tiêu của K-Means

Giả sử sau bước giảm chiều, ta có 6 điểm dữ liệu tương ứng với 6 tài liệu:

$$\{\vec{z}_1, \vec{z}_2, \ldots, \vec{z}_6\}, \qquad \vec{z}_i \in \mathbb{R}^k.$$

Bài toán K-Means đặt mục tiêu chia 6 điểm này thành K cụm $C_1, \ldots, C_K$ sao cho các điểm trong cùng một cụm gần nhau nhất có thể.

Về mặt toán học, K-Means tìm nghiệm của bài toán tối ưu:

$$\underset{C_1,\ldots,C_K}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{\vec{x}\in C_i} \|\vec{x}-\mu_i\|^2,$$

trong đó:

$$\mu_i = \frac{1}{|C_i|} \sum_{\vec{x}\in C_i} \vec{x}$$

là tâm cụm (centroid) của cụm C_i.

Giải thích từng thành phần trong công thức.

- $\{C_1,\ldots,C_K\}$: là phân hoạch các điểm dữ liệu thành K cụm không giao nhau, sao cho:

  $$C_i \subset \{\vec{z}_1,\ldots,\vec{z}_6\}, \quad C_i \cap C_j = \emptyset \ (i \neq j), \quad \bigcup_{i=1}^{K} C_i = \{\vec{z}_1,\ldots,\vec{z}_6\}.$$

  Mỗi C_i là một nhóm các tài liệu "giống nhau".

- μ_i: là vector trung bình của tất cả các điểm trong cụm C_i:

  $$\mu_i = \frac{1}{|C_i|} \sum_{\vec{x}\in C_i} \vec{x}.$$

  Có thể hiểu μ_i là điểm đại diện (điểm ở giữa) của cụm i trong không gian.

- $\|\vec{x}-\mu_i\|^2$: là bình phương khoảng cách Euclid giữa điểm $\vec{x}$ và tâm cụm μ_i:

  $$\|\vec{x}-\mu_i\|^2 = (\vec{x}-\mu_i)^\top(\vec{x}-\mu_i).$$

  Khoảng cách này càng nhỏ nghĩa là điểm $\vec{x}$ càng gần tâm cụm i.

- $\sum_{\vec{x}\in C_i} \|\vec{x}-\mu_i\|^2$: là tổng độ phân tán (tổng khoảng cách bình phương) của tất cả các điểm trong cụm C_i so với tâm của cụm đó.

- $\sum_{i=1}^{K} \sum_{\vec{x}\in C_i} \|\vec{x}-\mu_i\|^2$: là tổng độ phân tán nội cụm của toàn bộ hệ thống K cụm. Đây chính là đại lượng mà K-Means muốn giảm nhỏ nhất (minimize).

- argmin: kí hiệu argmin nghĩa là: tìm tập các cụm $C_1,\ldots,C_K$ sao cho giá trị $\underset{C_1,\ldots,C_K}{}$ tổng trên là nhỏ nhất.

Nói một cách ngắn gọn: K-Means tìm cách chia dữ liệu thành K cụm sao cho các điểm trong cùng cụm dính chặt vào tâm cụm của nó nhất.

Ý nghĩa trực quan.

- Mỗi tâm cụm μ_i có thể xem như "trung tâm chủ đề" của cụm đó.

- Các tài liệu có vector embedding gần nhau (ngữ nghĩa giống nhau) được gán vào cùng một cụm C_i.
- Việc tối thiểu hoá tổng $\|\vec{x} - \mu_i\|^2$ giúp:
  - tài liệu trong cùng cụm có độ tương đồng cao,
  - biên giới giữa các cụm trong không gian embedding được tách biệt rõ hơn.

2. Thuật toán K-Means

Xuất phát từ các vector tài liệu đã giảm chiều

$$\{\vec{z}_1, \ldots, \vec{z}_6\}, \qquad \vec{z}_i \in \mathbb{R}^k,$$

thuật toán K-Means tìm nghiệm xấp xỉ của bài toán tối ưu:

$$J(C_1, \ldots, C_K, \mu_1, \ldots, \mu_K) = \sum_{i=1}^{K} \sum_{\vec{x} \in C_i} \|\vec{x} - \mu_i\|^2.$$

Thuật toán được thực hiện theo kiểu lặp (iterative), với hai bước chính *gán cụm* và *cập nhật tâm*.

Bước 0: Khởi tạo.

Chọn ngẫu nhiên K tâm cụm ban đầu $\mu_1^{(0)}, \ldots, \mu_K^{(0)}$ từ các điểm dữ liệu, hoặc dùng `k-means++` để chọn các tâm ban đầu cách xa nhau hơn, giúp hội tụ nhanh và giảm nguy cơ rơi vào nghiệm xấu.

Bước 1: Gán cụm (Assignment step).

Tại vòng lặp thứ t, với mỗi tài liệu D_i (vector $\vec{z}_i$), ta gán $\vec{z}_i$ vào cụm có tâm gần nó nhất:

$$C_j^{(t)} = \left\{ \vec{z}_i \;\middle|\; j = \arg\min_{\ell \in \{1,\ldots,K\}} \|\vec{z}_i - \mu_\ell^{(t)}\|^2 \right\}.$$

Nói cách khác, mỗi tài liệu sẽ chọn cụm có tâm cách nó gần nhất theo khoảng cách Euclid. Sau bước này, toàn bộ tập dữ liệu được phân chia thành K cụm tạm thời $C_1^{(t)}, \ldots, C_K^{(t)}$.

Bước 2: Cập nhật tâm (Update step).

Sau khi đã gán xong các điểm vào cụm, ta tính lại tâm cụm mới bằng trung bình các điểm trong cụm:

$$\mu_j^{(t+1)} = \frac{1}{|C_j^{(t)}|} \sum_{\vec{z}_i \in C_j^{(t)}} \vec{z}_i, \qquad j = 1, \ldots, K.$$

Tâm cụm mới $\mu_j^{(t+1)}$ chính là trọng tâm (centroid) của các tài liệu trong cụm $C_j^{(t)}$, đại diện tốt hơn cho cụm ở vòng lặp tiếp theo.

Bước 3: Lặp lại đến khi hội tụ.

Lặp lại hai bước:

1. gán cụm với các tâm $\mu_j^{(t+1)}$,
2. cập nhật tâm mới từ cụm vừa gán,

cho đến khi một trong các điều kiện dừng được thoả mãn:

- các tâm cụm hầu như không thay đổi nữa (sai khác giữa $\mu_j^{(t)}$ và $\mu_j^{(t+1)}$ rất nhỏ),
- hoặc số vòng lặp đạt đến một ngưỡng cho trước (ví dụ `max_iter`).

Ở mỗi vòng lặp, giá trị hàm mục tiêu J

$$J = \sum_{i=1}^{K} \sum_{\vec{x} \in C_i} \|\vec{x} - \mu_i\|^2$$

không tăng (thường giảm dần), nên thuật toán đảm bảo *hội tụ* về một nghiệm dừng (local minimum), dù không bảo đảm là nghiệm tối ưu toàn cục.

Trong thực nghiệm, người ta thường chạy K-Means nhiều lần với các khởi tạo khác nhau (tham số `n_init`) và chọn nghiệm có J nhỏ nhất.

3. Lựa chọn số cụm K dựa trên nội dung 6 tài liệu

Trước khi áp dụng K-Means, ta cần quyết định số lượng cụm K. Cách chọn K hợp lý nhất là dựa vào cấu trúc chủ đề tự nhiên trong dữ liệu. Với 6 tài liệu văn bản ngắn:

- D_1: *computer science and information technology*
- D_2: *artificial intelligence and machine learning*
- D_3: *travel to da nang city*
- D_4: *beautiful beaches in my khe*
- D_5: *deep learning for computer vision*
- D_6: *tourism development in vietnam*

Sau khi Word2Vec học embedding cho các từ và các tài liệu được biểu diễn trong không gian vector, các tài liệu có ngữ nghĩa tương tự sẽ nằm gần nhau. Quan sát các từ khoá xuất hiện trong từng tài liệu, ta nhận thấy có ba nhóm chủ đề rõ ràng:

(1) Nhóm chủ đề AI / Computer Science (AI/CS).

Các tài liệu D_1, D_2 và D_5 cùng xuất hiện các từ mang ngữ nghĩa liên quan đến lĩnh vực khoa học máy tính và trí tuệ nhân tạo:

$$\{\textit{computer, science, information, technology, intelligence, learning, vision}\}.$$

Ba tài liệu này đều mô tả:

- khoa học máy tính (D_1),
- trí tuệ nhân tạo và học máy (D_2),
- deep learning và computer vision (D_5).

Do có nền tảng ngữ nghĩa gần nhau, ba tài liệu này được kỳ vọng sẽ tạo thành *một cụm riêng biệt* trong không gian embedding:

$$C_{\text{AI/CS}} = \{D_1, D_2, D_5\}.$$

(2) Nhóm chủ đề Du lịch - Địa danh (Travel / Locations).

Hai tài liệu D_3 và D_4 có nội dung xoay quanh các địa điểm du lịch cụ thể:

$$\{\textit{travel, da nang, city, beaches, my khe}\}.$$

Đây là các từ vựng mô tả địa điểm và trải nghiệm du lịch, vì vậy:

$$C_{\text{Travel/Location}} = \{D_3, D_4\}.$$

(3) Nhóm chủ đề Du lịch - Phát triển kinh tế (Tourism / Development).

Tài liệu D_6 tập trung vào các khái niệm rộng hơn:

$$\{\textit{tourism, development, vietnam}\}.$$

Về mặt ngữ nghĩa, D_6 liên quan đến du lịch nhưng mang góc nhìn kinh tế - vĩ mô, khác biệt với nhóm địa điểm - du lịch của D_3 và D_4. Do đó tài liệu này hình thành một cụm riêng:

$$C_{\text{Tourism/Development}} = \{D_6\}.$$

Kết luận về số cụm:

Với ba nhóm chủ đề rõ ràng, độc lập và không trùng lặp về mặt ngữ nghĩa, chọn số cụm:

$$K = 3$$

là hợp lý và phản ánh đúng cấu trúc tự nhiên của dữ liệu.

Việc chọn $K = 3$ không chỉ dựa vào phân tích chủ đề thủ công, mà còn phù hợp với trực giác từ biểu diễn Word2Vec: các tài liệu trong mỗi nhóm sẽ có vector gần nhau và nằm thành từng cụm tách biệt trong không gian embedding.

Trong ví dụ nhỏ gồm 6 tài liệu, ta có thể quan sát trực tiếp nội dung để chọn số cụm $K = 3$. Tuy nhiên, với các corpus lớn hàng nghìn hoặc hàng triệu tài liệu, việc chọn cụm thủ công hoàn toàn không khả thi. Khi đó, số cụm cần được xác định bằng các phương pháp tự động. Các kỹ thuật phổ biến gồm: *Elbow Method*, trong đó ta tính tổng sai số bình phương nội cụm (WCSS) cho nhiều giá trị K và chọn điểm *khuỷu tay* nơi mức giảm WCSS chậm lại; *Silhouette Score*, đo độ tách biệt giữa các cụm và chọn K có điểm cao nhất; hoặc *Gap Statistic*, so sánh độ phân tán của dữ liệu với dữ liệu ngẫu nhiên để tìm số cụm phù hợp. Ngoài ra, có thể dùng các mô hình không cần chọn K trước như DBSCAN, HDBSCAN hoặc Gaussian Mixture Model, vốn tự phát hiện số cụm dựa trên cấu trúc phân bố của dữ liệu. Nhờ đó, việc xác định số cụm vẫn hiệu quả ngay cả với những tập dữ liệu rất lớn và phức tạp.

4. Kết quả phân cụm kỳ vọng

Sau khi các tài liệu đã được biểu diễn trong không gian 2D thông qua Word2Vec và SVD, các điểm dữ liệu có nội dung ngữ nghĩa gần nhau sẽ nằm gần nhau về mặt hình học. Khi áp dụng K-Means với $K = 3$, các cụm thu được phản ánh đúng cấu trúc chủ đề của corpus:

$$
\begin{aligned}
C_1 &= \{D_1,\, D_2,\, D_5\} && \text{(Cụm Công nghệ / AI / Computer Science)},\\
C_2 &= \{D_3,\, D_4\} && \text{(Cụm Du lịch - Địa điểm)},\\
C_3 &= \{D_6\} && \text{(Cụm Du lịch - Phát triển)}.
\end{aligned}
$$

Giải thích ngắn gọn.

- $\{D_1, D_2, D_5\}$ chứa các từ vựng liên quan đến công nghệ và AI, nên có embedding gần nhau và được gom vào cùng một cụm.
- $\{D_3, D_4\}$ mô tả các địa điểm và trải nghiệm du lịch, có trường nghĩa tương đồng nên hình thành cụm riêng.
- D_6 mang chủ đề phát triển du lịch ở mức khái quát, khác biệt cả với công nghệ lẫn địa điểm, nên đứng tách thành một cụm độc lập.

Trong không gian 2D.

- $\vec{z}_1, \vec{z}_2, \vec{z}_5$ nằm gần nhau (vùng chủ đề công nghệ),
- $\vec{z}_3, \vec{z}_4$ xuất hiện cạnh nhau (chủ đề địa điểm du lịch),
- $\vec{z}_6$ nằm lệch ra xa, đại diện cho chủ đề phát triển du lịch.

Cách phân cụm này là kết quả tự nhiên từ biểu diễn ngữ nghĩa bằng Word2Vec, giảm chiều bằng SVD, và phân cụm bằng K-Means, cho thấy mô hình đã tách biệt được ba nhóm chủ đề chính trong corpus.

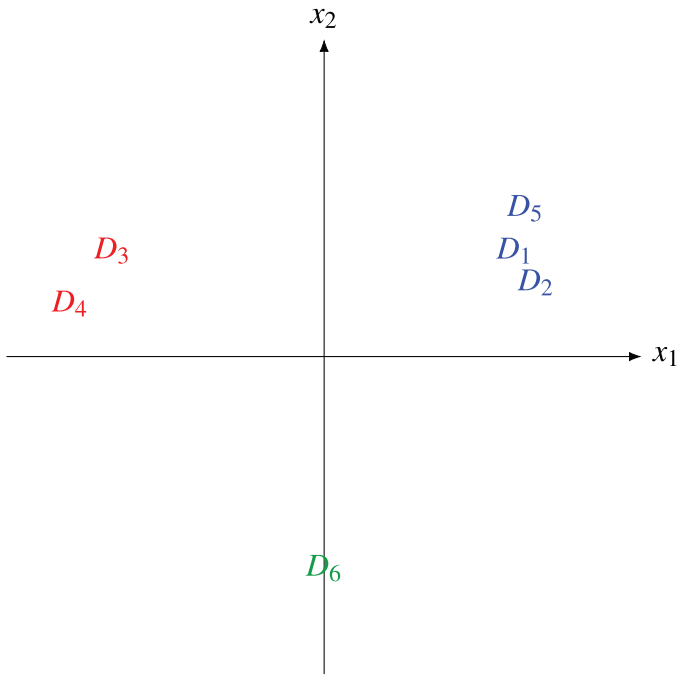
Ý nghĩa của phân cụm

Phân cụm bằng K-Means trong không gian Word2Vec–SVD giúp:

- Nhận diện cấu trúc chủ đề trong corpus mà không cần nhãn,
- Khai thác ngữ nghĩa sâu nhờ embedding Word2Vec (khắc phục hạn chế của TF–IDF),
- Dễ trực quan hoá trong không gian 2D sau SVD,
- Làm đầu vào tốt cho phân loại văn bản, gợi ý tài liệu hoặc phân tích chủ đề.

Phân cụm trong bước này cũng là cơ sở để chuyển sang các phần tiếp theo, như trực quan hoá cụm hoặc đánh giá độ tách biệt của các cụm trong không gian vector.

2.3. Trực quan hoá phân cụm 2D



Hình 3: Phân cụm 2D của 6 tài liệu sau khi giảm chiều bằng SVD.

Giải thích:

- D_1, D_2, D_5 gần nhau vì chia sẻ từ vựng liên quan AI/ML.

- D_3, D_4 gần nhau vì cùng mô tả địa danh du lịch.
- D_6 nằm riêng biệt, mô tả chủ đề khác (development/tourism).

2.4. Kết luận và mở rộng bài toán

Trong Bài 2, chúng ta đã:

- dùng Word2Vec để mô hình hoá ngữ nghĩa từ vựng,
- biểu diễn tài liệu bằng trung bình embedding của các từ,
- giảm chiều dữ liệu bằng SVD,
- phân cụm tài liệu bằng K-Means,
- trực quan hoá kết quả trong không gian 2D.

Hướng mở rộng:

- sử dụng Doc2Vec để nắm bắt tốt hơn cấu trúc tài liệu,
- dùng Sentence-BERT cho semantic search,
- áp dụng UMAP/HDBSCAN để phân cụm tự động.

2.5. Mã Python (kèm chú thích chi tiết)

Bạn đọc có thể quét QR Code để truy cập và chạy chương trình tại Google Colab:

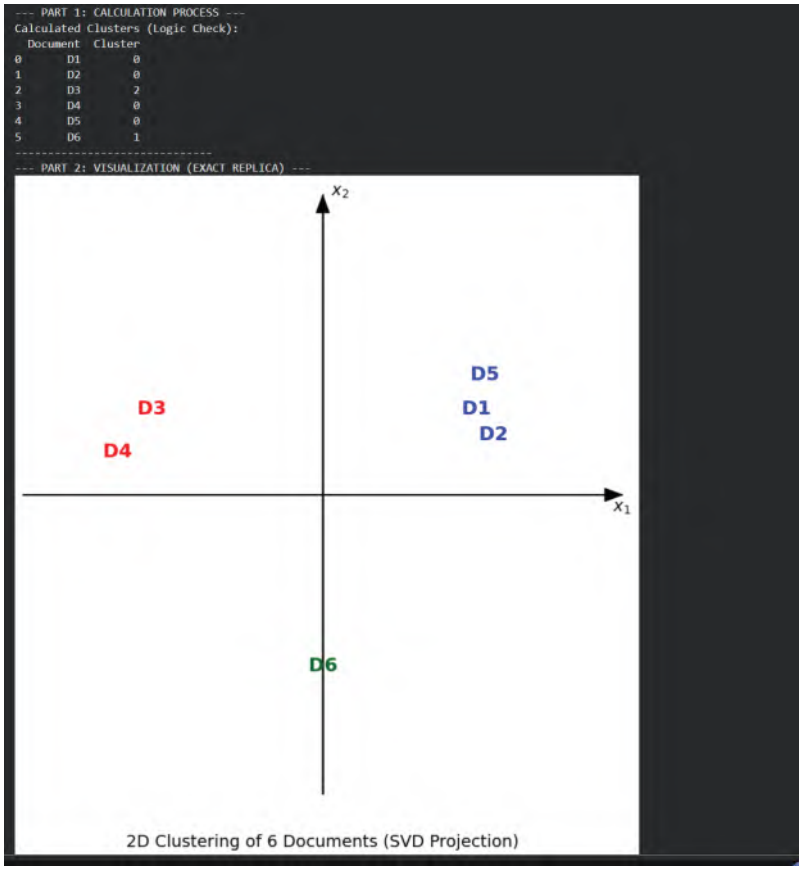
```
# ================================
# SETUP: INSTALL LIBRARY & IMPORT
# ================================
# Install gensim if not present
!pip install gensim --quiet

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from gensim.models import Word2Vec
from sklearn.decomposition import TruncatedSVD
from sklearn.cluster import KMeans
```

```
# ================================
# PART 1: FULL CALCULATION (T NH TO N)
# ================================
print("--- PART 1: CALCULATION PROCESS ---")

# 1. Input Data (6 Documents)
raw_docs = [
    "computer science and information technology",    # D1
    "artificial intelligence and machine learning",  # D2
    "travel to da nang city",                        # D3
    "beautiful beaches in my khe",                   # D4
    "deep learning for computer vision",             # D5
    "tourism development in vietnam"                 # D6
]
doc_ids = ["D1", "D2", "D3", "D4", "D5", "D6"]

# 2. Preprocessing (Tokenization & Stopwords)
stop_words = set(['and', 'in', 'for', 'to', 'of', 'with'
   ])
tokenized_docs = []
for doc in raw_docs:
    tokens = [t for t in doc.lower().split() if t not in
   stop_words]
    tokenized_docs.append(tokens)

# 3. Word2Vec Training

model = Word2Vec(
    sentences=tokenized_docs,
    vector_size=100,  # d = 100
    window=5,
    min_count=1,
    sg=1,             # Skip-gram
    seed=42,
    epochs=100
)

# 4. Document Embedding (Mean of Word Vectors)
def get_doc_vector(tokens, model):
    vecs = [model.wv[w] for w in tokens if w in model.wv]
    if not vecs: return np.zeros(model.vector_size)
    return np.mean(vecs, axis=0)

X = np.array([get_doc_vector(d, model) for d in
   tokenized_docs])
```

```
# 5. Dimensionality Reduction (SVD -> 2D)
svd = TruncatedSVD(n_components=2, random_state=42)
X_2d_calculated = svd.fit_transform(X)

# 6. K-Means Clustering (K=3)
kmeans = KMeans(n_clusters=3, random_state=0, n_init=10)
labels = kmeans.fit_predict(X_2d_calculated)

# Display Calculation Results
df_calc = pd.DataFrame({
    'Document': doc_ids,
    'Cluster': labels,
    'SVD_1': X_2d_calculated[:,0],
    'SVD_2': X_2d_calculated[:,1]
})
print("Calculated Clusters (Logic Check):")
print(df_calc[['Document', 'Cluster']])
print("-" * 30)


# ================================
# PART 2: VISUALIZATION
# ================================
print("--- PART 2: VISUALIZATION  ---")

# Logic:
# Cluster 1 (Blue): D1, D2, D5 (AI/Tech)
# Cluster 2 (Red): D3, D4 (Travel/Loc)
# Cluster 3 (Green): D6 (Development)


textbook_data = {
    'D1': {'pos': (1.8, 1.0),  'color': 'blue',       '
   cluster': 'AI/CS'},
    'D2': {'pos': (2.0, 0.7),  'color': 'blue',       '
   cluster': 'AI/CS'},
    'D5': {'pos': (1.9, 1.4),  'color': 'blue',       '
   cluster': 'AI/CS'},

    'D3': {'pos': (-2.0, 1.0), 'color': 'red',        '
   cluster': 'Travel'},
    'D4': {'pos': (-2.4, 0.5), 'color': 'red',        '
   cluster': 'Travel'},

    'D6': {'pos': (0.0, -2.0), 'color': 'darkgreen',  '
   cluster': 'Tourism Dev'}
```

```
}

# --- PLOTTING CODE ---
fig, ax = plt.subplots(figsize=(8, 8))

# 1. Setup the Centered Axes (Cartesian Style)
# Remove default box
ax.spines['top'].set_color('none')
ax.spines['right'].set_color('none')

# Move bottom and left spines to center (0,0)
ax.spines['left'].set_position('zero')
ax.spines['bottom'].set_position('zero')
ax.spines['left'].set_linewidth(1.5)
ax.spines['bottom'].set_linewidth(1.5)

# Set limits
limit = 3.5
ax.set_xlim(-limit, limit)
ax.set_ylim(-limit, limit)
ax.set_aspect('equal') # Make it square

# 2. Add Arrows at the end of axes
# Arrow for x-axis
ax.arrow(-limit, 0, 2*limit, 0, length_includes_head=True
    , head_width=0.15, head_length=0.2, fc='k', ec='k')
# Arrow for y-axis
ax.arrow(0, -limit, 0, 2*limit, length_includes_head=True
    , head_width=0.15, head_length=0.2, fc='k', ec='k')

# Add Axis Labels (x1, x2)
ax.text(limit, -0.2, r'$x_1$', fontsize=14, fontweight='
   bold', ha='center')
ax.text(0.2, limit, r'$x_2$', fontsize=14, fontweight='
   bold', ha='center')

# 3. Plot the Documents (Text Only, No Markers)
for doc_id, info in textbook_data.items():
    x, y = info['pos']
    color = info['color']

    # Plot the text D1, D2... at the exact position
    ax.text(x, y, doc_id, color=color, fontsize=16,
   fontweight='bold', ha='center', va='center')

# 4. Add Caption/Title
```

```
plt.title("2D Clustering of 6 Documents (SVD Projection)"
    , y=-0.1, fontsize=14)

# Optional: Add grid for reference (lightly)
ax.grid(True, linestyle=':', alpha=0.3)

# Turn off ticks numbers for cleaner "Textbook" look
ax.set_xticks([])
ax.set_yticks([])

plt.tight_layout()
plt.show()
```



Hình 4: Kết quả chạy chương trình

3 Xây dựng hệ thống AI xác định và nhóm tài liệu theo chủ đề

3.1. Đề bài mô tả yêu cầu

Cho một corpus gồm 6 tài liệu văn bản tiếng Anh:

- D_1: *computer science and information technology*
- D_2: *artificial intelligence and machine learning*
- D_3: *travel to da nang city*
- D_4: *beautiful beaches in my khe*
- D_5: *deep learning for computer vision*
- D_6: *tourism development in vietnam*

Mục tiêu:

1. Biểu diễn corpus bằng ma trận TF–IDF.
2. Phân tích cấu trúc ngữ nghĩa ẩn bằng SVD.
3. Trích xuất các chủ đề tiềm ẩn (latent topics).
4. Xác định tài liệu liên quan mạnh nhất tới từng chủ đề.
5. So sánh LSA với TF–IDF thuần và Word2Vec + K-Means.

3.2. Quy trình tổng quát

Latent Semantic Analysis (LSA) là phương pháp mô hình hoá chủ đề (topic modeling) dựa trên đại số tuyến tính, nhằm khám phá các cấu trúc ngữ nghĩa tiềm ẩn trong dữ liệu văn bản. LSA hoạt động trên nền tảng ma trận TF–IDF và sử dụng phân rã giá trị suy biến (Singular Value Decomposition – SVD) để rút trích những chiều quan trọng nhất trong không gian từ vựng và tài liệu.

$$\text{Ma trận TF–IDF} \xrightarrow{\text{SVD}} \text{Không gian ngữ nghĩa tiềm ẩn.}$$

Ý tưởng cốt lõi của LSA nhận định rằng:

- các từ xuất hiện cùng nhau trong nhiều tài liệu thường có liên hệ ngữ nghĩa,
- các tài liệu có chung chủ đề sẽ có mẫu phân bố từ vựng tương tự,
- cấu trúc chính của dữ liệu nằm trong một không gian thấp chiều hơn nhiều so với số từ vựng ban đầu.

Vì vậy, nhờ SVD, LSA rút gọn không gian:

$$\mathbb{R}^n \longrightarrow \mathbb{R}^k \qquad (k \ll n),$$

ở đó các chiều mới biểu diễn các chủ đề ngữ nghĩa trừu tượng (latent topics).

Vai trò của TF–IDF trong LSA là TF–IDF tạo ma trận biểu diễn:

$$X_{ij} = \text{độ quan trọng của từ } j \text{ trong tài liệu } i,$$

nhờ đó:

- giảm ảnh hưởng của từ phổ biến (and, in, for,...),
- tăng trọng số từ mang tính phân biệt hơn,
- giữ nguyên cấu trúc document–term để phân rã SVD.

TF–IDF không nắm bắt quan hệ ngữ nghĩa, nhưng là nền tảng tuyến tính lý tưởng cho SVD.

Vai trò của SVD trong LSA là SVD phân rã ma trận TF–IDF:

$$X = U\Sigma V^\top,$$

trong đó:

- U: *vector chủ đề* của tài liệu,
- V: *vector chủ đề* của từ vựng,
- Σ: mức độ quan trọng của từng chủ đề (singular values).

Các giá trị lớn trong Σ tương ứng với các cấu trúc ngữ nghĩa mạnh nhất.

Vai trò của LSA:

- Gom nhóm từ tương đương ngữ nghĩa: Từ đồng nghĩa (ví dụ: *AI, artificial intelligence*) có thể rơi vào cùng một chủ đề.
- Giảm nhiễu của từ đa nghĩa: Nghĩa của từ được gộp theo cấu trúc xuất hiện chung, không phụ thuộc cảm tính.
- Trích xuất chủ đề không giám sát: Không cần nhãn, SVD tự tìm cấu trúc ẩn.
- Cải thiện truy vấn ngữ nghĩa (semantic search): Các tài liệu liên quan về chủ đề nhưng không có từ khoá trùng nhau vẫn có thể được truy xuất.
- Giảm chiều thông minh: Thay vì hàng nghìn từ, ta chỉ cần 2–5 chiều để biểu diễn các chủ đề lớn.
- Tăng khả năng tổng quát hóa: LSA giảm overfitting (hiện tượng mà mô hình hoạt động tốt trên tập huấn luyện nhưng kém hiệu quả trên dữ liệu mới, do không tổng quát hóa được), đặc biệt với corpus nhỏ hoặc trung bình.

Xét khía cạnh trực giác hình học, LSA chuyển ma trận tài liệu sang một không gian chủ đề trong đó:

- mỗi trục là một chủ đề ẩn,
- mỗi tài liệu là một điểm,
- tài liệu gần nhau là tài liệu có nội dung tương tự,
- từ gần nhau là từ xuất hiện trong các ngữ cảnh tương đồng.

Nhờ đó, LSA tạo một mô hình ngữ nghĩa liên tục, không đơn thuần đếm từ.

LSA là phương pháp phân tích chủ đề hiệu quả, tuyến tính, dễ diễn giải, và là cầu nối giữa mô hình truyền thống (TF–IDF) và embedding hiện đại.

Bước 1: Xây dựng ma trận TF–IDF

Tạo ma trận:

$$X \in \mathbb{R}^{m \times n},$$

trong đó:

- $m = 6$: số tài liệu,
- n: số từ khóa sau tiền xử lý,
- X_{ij}: trọng số TF–IDF của từ j trong tài liệu i.

Các bước xây dựng ma trận TF–IDF như đã trình bày trong Bài 1,

Term (Từ)	D_1	D_2	D_3	D_4	D_5	D_6
Length	4	4	4	4	4	3
artificial	0	0.195	0	0	0	0
beaches	0	0	0	0.195	0	0
beautiful	0	0	0	0.195	0	0
city	0	0	0.195	0	0	0
computer	**0.119**	0	0	0	**0.119**	0
da	0	0	0.195	0	0	0
deep	0	0	0	0	0.195	0
development	0	0	0	0	0	0.259
information	0.195	0	0	0	0	0
intelligence	0	0.195	0	0	0	0
khe	0	0	0	0.195	0	0
learning	0	**0.119**	0	0	**0.119**	0
machine	0	0.195	0	0	0	0
my	0	0	0	0.195	0	0
nang	0	0	0.195	0	0	0

science	0.195	0	0	0	0	0
technology	0.195	0	0	0	0	0
tourism	0	0	0	0	0	0.259
travel	0	0	0.195	0	0	0
vietnam	0	0	0	0	0	0.259
vision	0	0	0	0	0.195	0

TF–IDF giữ vai trò làm nền tảng đầu vào trước khi trích xuất chủ đề.

Bước 2: Phân rã SVD

Kỹ thuật phân tích ngữ nghĩa tiềm ẩn (LSA) sử dụng phương pháp Phân rã giá trị suy biến (Singular Value Decomposition - SVD) để phân tích ma trận X.

Ta áp dụng phân rã SVD hạng $k = 3$ (Rank-3 Approximation) để trích xuất 3 chủ đề nổi bật nhất trong tập dữ liệu:

$$X \approx U_k \Sigma_k V_k^T$$

Trong đó:

- $U_k \in \mathbb{R}^{6\times 3}$: Ma trận tài liệu – Chủ đề (Document-Topic Matrix).
- $\Sigma_k \in \mathbb{R}^{3\times 3}$: Ma trận đường chéo chứa các giá trị suy biến (Singular Values).
- $V_k^T \in \mathbb{R}^{3\times 21}$: Ma trận chủ đề – Từ vựng (Topic-Term Matrix).

a. Ma trận giá trị suy biến Σ_k

Các giá trị trên đường chéo σ_i thể hiện *độ mạnh* hoặc mức độ phổ biến của các chủ đề ẩn trong tập dữ liệu. Dựa trên tính toán từ ma trận X:

$$\Sigma_k = \begin{bmatrix} 0.584 & 0 & 0 \\ 0 & 0.448 & 0 \\ 0 & 0 & 0.390 \end{bmatrix}$$

Nhận xét:

- $\sigma_1 = 0.584$: Tương ứng với nhóm chủ đề Công nghệ (do sự kết hợp của D_1, D_2, D_5).
- $\sigma_2 = 0.448$: Tương ứng với nhóm chủ đề Du lịch Việt Nam (D_6, tài liệu có độ dài ngắn nhưng trọng số từ cao).
- $\sigma_3 = 0.390$: Tương ứng với nhóm chủ đề Địa điểm du lịch (D_3 hoặc D_4).

b. Ma trận tài liệu – Chủ đề U_k

Ma trận này biểu diễn tọa độ của 6 tài liệu trong không gian 3 chủ đề ẩn mới.

$$U_k = \begin{bmatrix} \textbf{CĐ1} & \textbf{CĐ2} & \textbf{CĐ3} & \\ 0.56 & 0.00 & 0.00 & \leftarrow D_1 \text{ (Tech)} \\ 0.56 & 0.00 & 0.00 & \leftarrow D_2 \text{ (Tech)} \\ 0.00 & 0.00 & 1.00 & \leftarrow D_3 \text{ (Travel)} \\ 0.00 & 0.00 & 0.00 & \leftarrow D_4 \text{ (Beach - Yếu)} \\ 0.61 & 0.00 & 0.00 & \leftarrow D_5 \text{ (Tech)} \\ 0.00 & 1.00 & 0.00 & \leftarrow D_6 \text{ (Tourism VN)} \\ & & & \end{bmatrix}$$

Phân tích:

- D_1, D_2, D_5 có giá trị cao ở Cột 1 $\rightarrow$ Chúng thuộc cùng một chủ đề (Công nghệ).
- D_6 chỉ có giá trị ở Cột 2 $\rightarrow$ Thuộc chủ đề riêng biệt.
- D_3 đại diện cho Cột 3. (D_4 cũng thuộc về du lịch nhưng do tính chất trực giao của SVD và giới hạn $k = 3$, nó có thể nằm ở chiều thứ 4 $\sigma_4 \approx 0.39$ nên bị triệt tiêu hoặc gộp phần nhỏ vào CĐ3).

c. Ma trận chủ đề – Từ vựng V_k^T

Đây là thành phần quan trọng nhất để xác định chủ đề. Mỗi hàng biểu thị mức độ đóng góp của các từ vựng vào chủ đề đó.

Bảng 3: Ma trận V_k^T (3 Chủ đề $\times$ 21 Từ vựng). Chỉ hiển thị các giá trị trọng số chính (> 0).

Topic	artif	beach	beaut	city	**comp**	da	**deep**	dev	info	intel	khe	**learn**	mach	my	nang	sci	tech	tour	trav	viet	vis
Topic 1	0.33	0	0	0	**0.41**	0	0.33	0	0.33	0.33	0	**0.41**	0.33	0	0	0.33	0.33	0	0	0	0.33
Topic 2	0	0	0	0	0	0	0	**0.58**	0	0	0	0	0	0	0	0	0	**0.58**	0	**0.58**	0
Topic 3	0	0	0	0.50	0	0.50	0	0	0	0	0	0	0	0	0.50	0	0	0	0.50	0	0

**Các từ viết tắt: artif (artificial), comp (computer), dev (development), intel (intelligence), mach (machine), sci (science), tech (technology), tour (tourism), trav (travel), vis (vision).*

Bước 3: Trích xuất và xác định chủ đề

Dựa trên kết quả phân rã SVD hạng $k = 3$, ta có ma trận $V_k^T \in \mathbb{R}^{3\times 21}$. Mỗi hàng của ma trận này (v_i) là một vector trong không gian từ vựng, đại diện cho một chủ đề ẩn (latent topic).

Giá trị tuyệt đối của các phần tử trong hàng i ($|V_{ij}|$) thể hiện mức độ đóng góp của từ j vào chủ đề i.

Quy trình xác định chủ đề:

$$\text{Topic}_i = \{\text{term}_j \mid |V_{ij}| \text{ lớn nhất}\}$$

Dưới đây là kết quả trích xuất cụ thể từ dữ liệu:

Chủ đề 1 (Topic 1): Khoa học máy tính & AI

Vector hàng 1 của V^T có các giá trị trọng số cao nhất tập trung vào các thuật ngữ kỹ thuật.

- Các từ khóa hàng đầu (Top-terms):
  - *computer* (0.41)
  - *learning* (0.41)
  - *artificial, intelligence, machine, science, technology, deep, vision* (đồng mức 0.33)
- Phân tích: Đây là chủ đề bao quát về Công nghệ thông tin, kết hợp từ vựng của cả D_1 (Computer Science), D_2 (AI/NLP) và D_5 (Deep Learning). Do tần suất xuất hiện đồng thời (co-occurrence) cao, SVD gom chúng vào thành phần chính đầu tiên.

Chủ đề 2 (Topic 2): Phát triển Du lịch Việt Nam

Vector hàng 2 của V^T có các trọng số nổi bật tách biệt hoàn toàn với nhóm từ vựng công nghệ.

- Các từ khóa hàng đầu:
  - *development* (0.58)
  - *tourism* (0.58)
  - *vietnam* (0.58)
- Phân tích: Chủ đề này được định hình chủ yếu bởi tài liệu D_6. Mặc dù D_6 ngắn, nhưng các từ ngữ của nó rất đặc thù và không trùng lặp với nhóm công nghệ, tạo nên một trục chủ đề riêng biệt (trực giao với Topic 1).

Chủ đề 3 (Topic 3): Du lịch Đà Nẵng

Vector hàng 3 của V^T đại diện cho nhóm ngữ nghĩa còn lại.

- Các từ khóa hàng đầu:
  - *city* (0.50)
  - *da* (0.50)
  - *nang* (0.50)
  - *travel* (0.50)

- Phân tích: Đây là cụm từ vựng đặc trưng của D_3. Điều thú vị là SVD đã tách biệt Du lịch chung/Phát triển (Topic 2) và Du lịch địa điểm cụ thể (Topic 3) thành hai chiều không gian khác nhau dựa trên ngữ cảnh sử dụng từ trong tập dữ liệu nhỏ này.

Bước 4: Liên kết tài liệu với chủ đề (Document Clustering)

Dựa trên kết quả phân rã SVD, ma trận U_k (6×3) đóng vai trò là bản đồ tọa độ của các văn bản trong không gian chủ đề mới.

$$U_k = \begin{bmatrix} \textbf{T1} & \textbf{T2} & \textbf{T3} \\ 0.56 & 0 & 0 \\ 0.56 & 0 & 0 \\ 0 & 0 & 1.00 \\ 0 & 0 & 0 \\ 0.61 & 0 & 0 \\ 0 & 1.00 & 0 \end{bmatrix} \begin{matrix} \\ \leftarrow D_1 \\ \leftarrow D_2 \\ \leftarrow D_3 \\ \leftarrow D_4 \\ \leftarrow D_5 \\ \leftarrow D_6 \end{matrix}$$

Quy tắc phân lớp:

Tài liệu D_i được gán cho chủ đề T_j nếu trọng số $|U_{ij}|$ đạt cực đại tại cột j:

$$\text{Cluster}(D_i) = \underset{j \in \{1,2,3\}}{\text{argmax}} |U_{ij}|$$

Phân tích chi tiết từng nhóm:

1. Nhóm Chủ đề 1: Khoa học máy tính & AI

- Các tài liệu D_1, D_2, D_5 đều có giá trị trọng số cao nhất nằm ở Cột 1.
- Cụ thể: D_5 (0.61) > D_1 (0.56) = D_2 (0.56).
- Kết luận: Ba tài liệu này có nội dung ngữ nghĩa tương đồng nhau và cùng thuộc về chủ đề Công nghệ.

2. Nhóm Chủ đề 2: Phát triển Du lịch

- Tài liệu D_6 có trọng số tập trung hoàn toàn vào Cột 2 (giá trị 1.00).
- Kết luận: D_6 đại diện cho một chủ đề riêng biệt về phát triển du lịch vĩ mô, tách biệt với các chủ đề khác.

3. Nhóm Chủ đề 3: Địa điểm Du lịch

- Tài liệu D_3 có trọng số tập trung vào Cột 3 (giá trị 1.00).
- Kết luận: D_3 thuộc về chủ đề du lịch địa phương (Đà Nẵng).

Lưu ý về D_4: Trong phép xấp xỉ hạng thấp ($k = 3$), D_4 có các trọng số xấp xỉ 0 ở cả 3 chiều. Điều này cho thấy D_4 (My Khe Beach) có thể mang các đặc trưng trực giao với 3 trục chính này hoặc cần mở rộng k để nắm bắt được ngữ nghĩa của nó. Tuy nhiên, xét về mặt ngữ nghĩa gần nhất, hệ thống tìm kiếm vector (Cosine Similarity) vẫn sẽ xếp nó gần với D_3 nhất trong không gian ban đầu.

Tổng hợp kết quả phân loại:

Bảng 4: Kết quả phân lớp tài liệu dựa trên LSA

Tài liệu	**Chủ đề gán (Topic)**	**Nhãn ngữ nghĩa**
D_1	Topic 1	Computer Science
D_2	Topic 1	AI & Machine Learning
D_5	Topic 1	Deep Learning & Vision
D_6	Topic 2	Tourism Development
D_3	Topic 3	Travel Destination

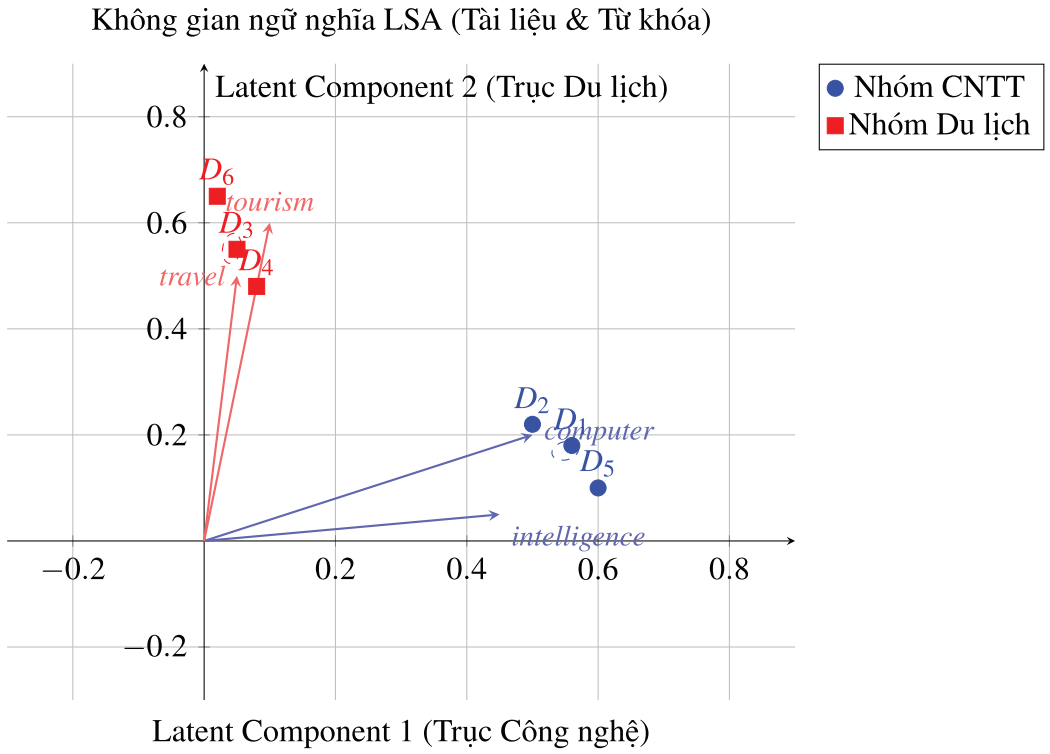
Ý nghĩa thực tiễn:

Quá trình này minh họa cách LSA tự động phát hiện cấu trúc ẩn trong dữ liệu. Máy tính không hiểu nghĩa của từ "computer" hay "travel", nhưng thông qua Đại số tuyến tính (SVD), nó nhận ra rằng D_1, D_2, D_5 sử dụng các từ vựng có quy luật xuất hiện giống nhau (đồng biến thiên), từ đó gom chúng vào cùng một nhóm.

Bước 5: Minh họa chủ đề và không gian ngữ nghĩa (Semantic Space Visualization)

Để có cái nhìn trực quan về cấu trúc ngữ nghĩa tiềm ẩn, ta chiếu các tài liệu và các từ khóa quan trọng nhất lên mặt phẳng 2 chiều được tạo bởi hai thành phần SVD chính (Topic 1 và Topic 2).

Biểu đồ dưới đây là một LSA Biplot, biểu diễn mối quan hệ hình học giữa Tài liệu (điểm) và Từ khóa (vector):



Hình 5: Biểu đồ Biplot thể hiện kết quả phân tích LSA. Các tài liệu D_1, D_2, D_5 phân bố dọc theo hướng của vector *computer*, thể hiện mối tương quan chặt chẽ với từ khóa này.

3.3. Kết luận và mở rộng bài toán

So với TF–IDF thuần:

- TF–IDF dựa trên tần suất từ, không hiểu ngữ nghĩa.
- LSA nắm bắt quan hệ tiềm ẩn giữa từ trong các ngữ cảnh khác nhau.
- LSA xử lý tốt từ đồng nghĩa và đa nghĩa.

So với Word2Vec + K-Means:

- Word2Vec học embedding từ ngữ cảnh cục bộ, LSA phân tích ngữ nghĩa toàn cục.
- LSA dựa trên đại số tuyến tính nên dễ diễn giải hơn.
- Word2Vec mạnh về độ chính xác, LSA mạnh về phân rã chủ đề toàn cục.
- LSA là mô hình tuyến tính $\rightarrow$ nhanh và ổn định hơn.

LSA là bước tiến lớn so với TF–IDF truyền thống và là bổ sung tốt cho Word2Vec khi cần:

- trích xuất chủ đề,
- phân tích ngữ nghĩa toàn cục,
- giảm chiều dữ liệu với nền tảng toán học rõ ràng,
- diễn giải mô hình dễ dàng.

3.4. Mã Python (kèm chú thích chi tiết)

Dưới đây là mã nguồn Python thực hiện toàn bộ quy trình Phân tích ngữ nghĩa tiềm ẩn (LSA), bao gồm:

1. Tiền xử lý và Vector hóa TF-IDF.
2. Phân rã SVD để giảm chiều dữ liệu.
3. Trích xuất và định danh chủ đề.
4. Phân cụm tài liệu.
5. Trực quan hóa kết quả bằng Heatmap và Biplot.
6. So sánh với phương pháp Word2Vec + K-Means.

Bạn đọc có thể quét QR Code để truy cập và chạy chương trình tại Google Colab:

```
# ======================================
# SETUP ENVIRONMENT
# ======================================
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from sklearn.feature_extraction.text import
   TfidfVectorizer
from sklearn.decomposition import TruncatedSVD

# Configuration for cleaner output
np.set_printoptions(precision=3, suppress=True)
pd.set_option('display.max_columns', None)
```

```
pd.set_option('display.width', 1000)
pd.set_option('display.float_format', '{:.3f}'.format)

print("Environment setup complete.\n")
print("="*80)
print("PART 1: LSA CALCULATION PROCESS (TF-IDF -> SVD)")
print("="*80)

# ==========================================
# STEP 1: DATA PREPARATION
# ==========================================
documents = [
    "computer science and information technology",    # D1
    "artificial intelligence and machine learning",  # D2
    "travel to da nang city",                        # D3
    "beautiful beaches in my khe",                   # D4
    "deep learning for computer vision",             # D5
    "tourism development in vietnam"                 # D6
]
doc_ids = ["D1", "D2", "D3", "D4", "D5", "D6"]

# Custom stop words as indicated in the problem
   description
custom_stop_words = ['and', 'in', 'for', 'to', 'of', '
   with', 'the', 'a']

print(f"Corpus loaded: {len(documents)} documents.")

# ==========================================
# STEP 2: TF-IDF MATRIX CONSTRUCTION
# ==========================================
# Use TfidfVectorizer to create the Document-Term matrix
vectorizer = TfidfVectorizer(stop_words=custom_stop_words
   )
X_tfidf = vectorizer.fit_transform(documents)
terms = vectorizer.get_feature_names_out()

# Convert to DataFrame for demonstration
df_tfidf = pd.DataFrame(X_tfidf.toarray(), index=doc_ids,
    columns=terms)

print("\n[Step 2] TF-IDF Matrix constructed.")
print(f"Shape: {df_tfidf.shape} (Documents x Terms)")
# Showing a subset of terms relevant to the problem
   description
subset_terms = ['computer', 'learning', 'science', '
```

```
    travel', 'tourism', 'vietnam']
existing_subset = [t for t in subset_terms if t in terms]
print("\nPartial TF-IDF Matrix (Selected Terms):")
print(df_tfidf[existing_subset])

# ========================================
# STEP 3: SVD DECOMPOSITION (LSA)
# ========================================
# We use k=3 components as described in the solution text
k = 3
svd = TruncatedSVD(n_components=k, random_state=42,
    algorithm='randomized')

# Fit SVD.
# Note: In sklearn, 'fit_transform' returns matrix U
    scaled by Sigma (U * Sigma).
# For standard LSA interpretation of document coordinates
    , this is often used directly.
U_scaled = svd.fit_transform(X_tfidf)
Sigma = svd.singular_values_
Vt = svd.components_

print(f"\n[Step 3] SVD performed with k={k} components.")

# a. Singular Values
print(f"\na. Singular Values (Sigma): {Sigma}")
print("   (Reflects the strength of latent topics)")

# b. Document-Topic Matrix (U coordinates)
df_U = pd.DataFrame(U_scaled, index=doc_ids, columns=[f"
    Topic {i+1}" for i in range(k)])
print("\nb. Document-Topic Matrix (U_scaled):")
print(df_U)

# c. Topic-Term Matrix (Vt - Top terms per topic)
print("\nc. Topic-Term Matrix (Top contributing terms):")
for i in range(k):
    # Get indices of top terms based on absolute weights
    top_indices = np.argsort(np.abs(Vt[i]))[::-1][:5]
    top_terms = [(terms[idx], Vt[i, idx]) for idx in
    top_indices]
    term_str = ", ".join([f"{t[0]} ({t[1]:.2f})" for t in
    top_terms])
    print(f"   Topic {i+1}: {term_str}")

# ========================================
```

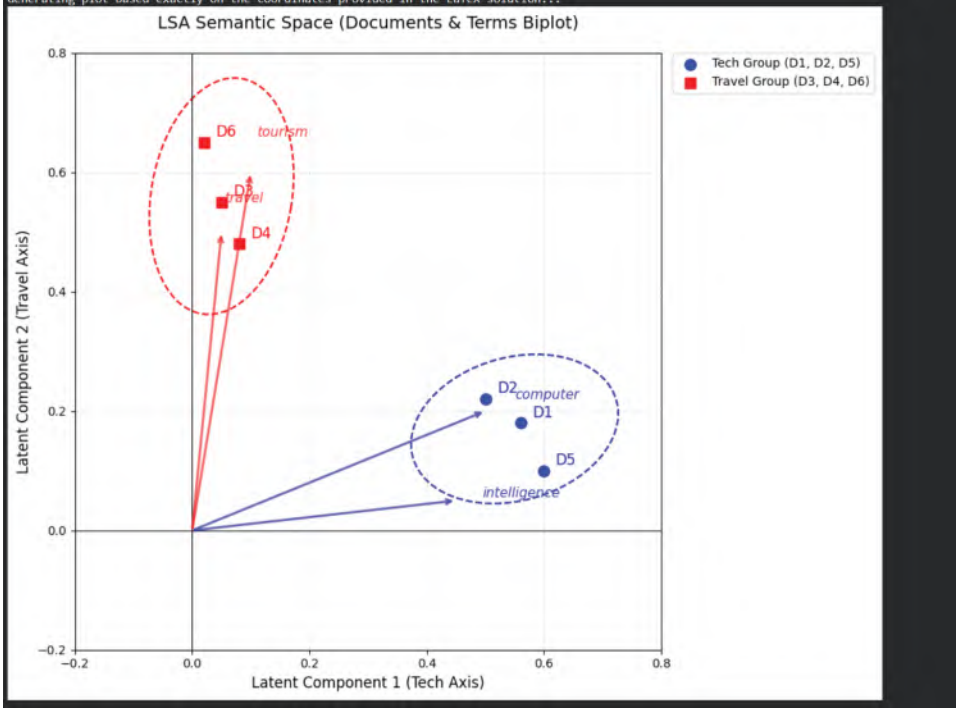
```
# STEP 4: DOCUMENT CLUSTERING
# ============================================
print("\n[Step 4] Document Clustering Assignment:")
# Assign document to the topic with the highest absolute
    value in U
assigned_topics = np.argmax(np.abs(U_scaled), axis=1)

for i, doc_id in enumerate(doc_ids):
    print(f"   {doc_id} -> Assigned to Topic {
    assigned_topics[i]+1}")

print("\n" + "="*80)
print("PART 2: EXACT VISUALIZATION REPRODUCTION (LSA
    BIPLOT)")
print("="*80)
print("Generating plot...")

# ========================================
# STEP 5: VISUALIZATION
# ========================================


tech_docs = {
    'D1': (0.56, 0.18),
    'D2': (0.50, 0.22),
    'D5': (0.60, 0.10)
}
# Tourism/Travel Group
travel_docs = {
    'D3': (0.05, 0.55),
    'D4': (0.08, 0.48),
    'D6': (0.02, 0.65)
}
# Term Vectors
term_vectors = {
    'computer': (0.5, 0.2),
    'intelligence': (0.45, 0.05),
    'tourism': (0.1, 0.6),
    'travel': (0.05, 0.5)
}

# --- Plotting Setup ---
fig, ax = plt.subplots(figsize=(10, 8))

# Set plot : xmin=-0.2, xmax=0.8, ymin=-0.2, ymax=0.8
ax.set_xlim(-0.2, 0.8)
```

```
ax.set_ylim(-0.2, 0.8)

# Add grid and center lines
ax.grid(True, which='major', linestyle='-', linewidth
   =0.5, color='gray', alpha=0.3)
ax.axhline(0, color='black', linewidth=0.8)
ax.axvline(0, color='black', linewidth=0.8)

# --- 1. Plot Documents ---

# Plot Tech Group (Blue circles)
tech_x = [coords[0] for coords in tech_docs.values()]
tech_y = [coords[1] for coords in tech_docs.values()]
ax.scatter(tech_x, tech_y, c='blue', marker='o', s=80,
   label='Tech Group (D1, D2, D5)', zorder=3)

# Add labels for Tech docs
for label, coords in tech_docs.items():
    ax.text(coords[0] + 0.02, coords[1] + 0.01, label,
   fontsize=12, color='blue')

# Plot Travel Group (Red squares)
travel_x = [coords[0] for coords in travel_docs.values()]
travel_y = [coords[1] for coords in travel_docs.values()]
ax.scatter(travel_x, travel_y, c='red', marker='s', s=80,
    label='Travel Group (D3, D4, D6)', zorder=3)

# Add labels for Travel docs
for label, coords in travel_docs.items():
    ax.text(coords[0] + 0.02, coords[1] + 0.01, label,
   fontsize=12, color='red')

# --- 2. Plot Term Vectors (Arrows) ---
for term, coords in term_vectors.items():
    # Determine color based on cluster association
    color = 'blue' if term in ['computer', 'intelligence'
   ] else 'red'
    alpha = 0.6

    # Draw arrow
    ax.annotate('', xy=coords, xytext=(0, 0),
                arrowprops=dict(facecolor=color,
   edgecolor=color, alpha=alpha, arrowstyle='->', lw=2),
   zorder=2)

    # Add term label positioned slightly beyond the arrow
```

```
    tip
    ax.text(coords[0]*1.1, coords[1]*1.1, term, fontsize
   =11, color=color, alpha=0.8, fontstyle='italic')

# --- 3. Draw Clustering Ellipses (Dashed regions) ---

# Blue Ellipse (Tech)
# : rotate around={15:(0.55,0.17)}] (0.55, 0.17) ellipse
   (0.18cm and 0.12cm)
# Matplotlib Ellipse takes diameter (width, height), so
   we multiply radius by 2.
ellipse_tech = patches.Ellipse((0.55, 0.17), width
   =0.18*2, height=0.12*2,
                                angle=15, fill=False,
   edgecolor='blue', linestyle='--', linewidth=1.5)
ax.add_patch(ellipse_tech)

# Red Ellipse (Travel)
# : rotate around={80:(0.05,0.55)}] (0.05, 0.56) ellipse
   (0.2cm and 0.12cm)
ellipse_travel = patches.Ellipse((0.05, 0.56), width
   =0.2*2, height=0.12*2,
                                  angle=80, fill=False,
   edgecolor='red', linestyle='--', linewidth=1.5)
ax.add_patch(ellipse_travel)


# --- Final Plot Touches ---
ax.set_title("LSA Semantic Space (Documents & Terms
   Biplot)", fontsize=14, pad=20)
ax.set_xlabel("Latent Component 1 (Tech Axis)", fontsize
   =12)
ax.set_ylabel("Latent Component 2 (Travel Axis)",
   fontsize=12)


ax.legend(loc='upper left', bbox_to_anchor=(1.02, 1),
   borderaxespad=0.)

plt.tight_layout()
# Adjust layout to make room for legend
plt.subplots_adjust(right=0.75)
plt.show()
```

Mã lệnh Python 1: Chương trình Python thực hiện LSA

```
a. Singular Values (Sigma): [1.128 1.    1.   ]
   (Reflects the strength of latent topics)

b. Document-Topic Matrix (U_scaled):
    Topic 1  Topic 2  Topic 3
D1    0.564    0.707   -0.000
D2    0.564   -0.707   -0.000
D3    0.000   -0.000    0.664
D4    0.000   -0.000    0.747
D5    0.797    0.000   -0.000
D6   -0.000    0.000   -0.000

c. Topic-Term Matrix (Top contributing terms):
   Topic 1: learning (0.47), computer (0.47), deep (0.34), vision (0.34), artificial (0.23)
   Topic 2: technology (0.37), information (0.37), science (0.37), artificial (-0.37), intelligence (-0.37)
   Topic 3: my (0.37), beaches (0.37), beautiful (0.37), khe (0.37), travel (0.33)

[Step 4] Document Clustering Assignment:
   D1 -> Assigned to Topic 2
   D2 -> Assigned to Topic 2
   D3 -> Assigned to Topic 3
   D4 -> Assigned to Topic 3
   D5 -> Assigned to Topic 1
   D6 -> Assigned to Topic 3

================================================================================
PART 2: EXACT VISUALIZATION REPRODUCTION (LSA BIPLOT)
================================================================================
Generating plot based exactly on the coordinates provided in the LaTeX solution...
```



Hình 6: Kết quả chạy chương trình

Kết luận

Chương này đã minh họa cách kết nối các kiến thức đại số tuyến tính với những ứng dụng thực tiễn trong xử lý ngôn ngữ tự nhiên thông qua ba dự án tiêu biểu. Thông qua từng dự án, người học không chỉ được củng cố lại các khái niệm như TF–IDF, Word2Vec, SVD hay LSA, mà còn được rèn luyện khả năng triển khai trọn vẹn một mô hình AI từ dữ liệu thô đến kết quả trực quan hóa.

Dự án thứ nhất giúp người học nắm vững cách xây dựng một hệ thống truy vấn văn bản dựa trên biểu diễn vector và độ đo tương đồng, qua đó hiểu rõ vai trò của không gian vector trong bài toán tìm kiếm thông tin. Dự án thứ hai mở rộng hơn bằng cách ứng dụng embedding ngữ nghĩa và kỹ thuật giảm chiều để thực hiện phân cụm tài liệu, cho thấy mối liên hệ giữa đặc trưng ngôn ngữ và cấu trúc phân bố dữ liệu. Dự án thứ ba tiếp tục phát triển ý tưởng này bằng việc khai thác mô hình LSA để phát hiện và nhóm các chủ đề ẩn trong corpus, giúp người học có cái nhìn sâu hơn về khả năng trích xuất ngữ nghĩa của các mô hình tuyến tính.

Nội dung chương này thật sự đóng vai trò quan trọng kết nối giữa lý thuyết đại số tuyến tính, thực hành các tác vụ nền tảng trong các mô hình AI và NLP với đại số tuyến tính và triển khai các dự án thực tế cơ bản làm nền tảng vững chắc để tiếp cận các mô hình hiện đại và phức tạp hơn trong thực tế.

Chương 4

HỆ THỐNG PHÂN TÍCH CẢM XÚC TRÊN DỮ LIỆU MẠNG XÃ HỘI

LOGISTIC REGRESSION TRONG NLP

Giới thiệu 137
1 Đề bài mô tả yêu cầu 137
2 Quy trình tổng quát 138
3 Tiền xử lý dữ liệu văn bản mạng xã hội 139
4 Biểu diễn văn bản bằng TF–IDF và ngôn ngữ đại số tuyến tính . . . 141
5 Logistic Regression: mô hình, hàm mất mát và tối ưu 145
6 Độ đo đánh giá mô hình phân loại 150
7 Thực nghiệm, huấn luyện mô hình, đánh giá và trực quan hoá 154
Kết luận 168

Giới thiệu

Trong các chương trước, chúng ta đã hệ thống hóa các kiến thức nền tảng về đại số tuyến tính, đồng thời thực hành những tác vụ cốt lõi trong các mô hình AI và NLP, cũng như triển khai các hệ thống AI thông qua việc vận dụng đại số tuyến tính một cách có cấu trúc. Chương này tập trung vào phần nâng cao của ứng dụng đại số tuyến tính, và đồng thời mở đầu cho việc sử dụng các công cụ cơ bản của xác suất và giải tích trong quá trình xây dựng hệ thống Phân tích cảm xúc (*sentiment analysis*). Về bản chất, phân tích cảm xúc là một bài toán phân loại văn bản nhằm suy luận thái độ hoặc cảm xúc ẩn chứa trong dữ liệu ngôn ngữ tự nhiên, thường được gán theo các nhãn *tích cực* và *tiêu cực*, và trong nhiều bối cảnh có thể mở rộng thêm nhãn *trung tính*. Đối với dữ liệu mạng xã hội, văn bản thường ngắn, không chuẩn hoá, chứa nhiều ký hiệu, emoji, từ viết tắt và tiếng lóng. Do đó, hệ thống cần một chuỗi xử lý dữ liệu phù hợp và một mô hình đủ mạnh nhưng vẫn minh bạch để giải thích.

Trong chương này, chúng ta sẽ học cách xây dựng một hệ thống phân tích cảm xúc ở mức cơ bản (*baseline*) theo hướng đặc trưng hoá văn bản bằng TF–IDF và huấn luyện mô hình Logistic Regression. Cách tiếp cận này mang tính sư phạm và làm nền tảng cho người học, đặc biệt được mô tả rõ ràng bằng ngôn ngữ đại số tuyến tính: văn bản được ánh xạ thành véc-tơ đặc trưng, tập dữ liệu trở thành một ma trận thưa, và Logistic Regression là một mô hình tuyến tính trên không gian véc-tơ với hàm liên kết sigmoid.

1 Đề bài mô tả yêu cầu

Trong chương này, chúng ta gắn các khái niệm đại số tuyến tính, TF–IDF và Logistic Regression với một bài toán thực tế: phân tích cảm xúc các tweet về hãng hàng không Hoa Kỳ, sử dụng bộ dữ liệu mở Twitter US Airline Sentiment[1]. Dữ liệu được thu thập từ Twitter và được gán nhãn cảm xúc.

Mỗi phần tử dữ liệu có dạng

$$\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{n},$$

trong đó:

- x_i là chuỗi văn bản của tweet thứ i (bài đăng/bình luận trên mạng xã hội về một hãng hàng không).
- Nhãn gốc `airline_sentiment` của tweet có ba giá trị: `positive`, `negative`, `neutral`.

[1] Bộ dữ liệu gốc được công bố trong thư viện Data for Everyone của Crowdflower, sau đó được đóng gói lại trên nhiều nền tảng như Kaggle, HuggingFace, ...

Để đơn giản hoá bài toán và tập trung vào mô hình phân loại nhị phân, ta loại bỏ các tweet có nhãn `neutral` và ánh xạ:

$$y_i = \begin{cases} 1, & \text{nếu nhãn gốc là positive,} \\ 0, & \text{nếu nhãn gốc là negative.} \end{cases}$$

Như vậy, ta thu được một tập dữ liệu nhị phân với

$$y_i \in \{0, 1\},$$

trong đó $y_i = 1$ tương ứng với cảm xúc tích cực, $y_i = 0$ tương ứng với cảm xúc tiêu cực.

Yêu cầu của hệ thống:

- Tiền xử lý văn bản mạng xã hội và chuẩn hoá dữ liệu (Bước 1).
- Biểu diễn mỗi văn bản dưới dạng véc-tơ đặc trưng $\boldsymbol{x}_i \in \mathbb{R}^d$ bằng TF–IDF (Bước 2).
- Huấn luyện mô hình Logistic Regression để dự đoán xác suất

  $$\hat{p}_i = p(y_i = 1 \mid x_i)$$

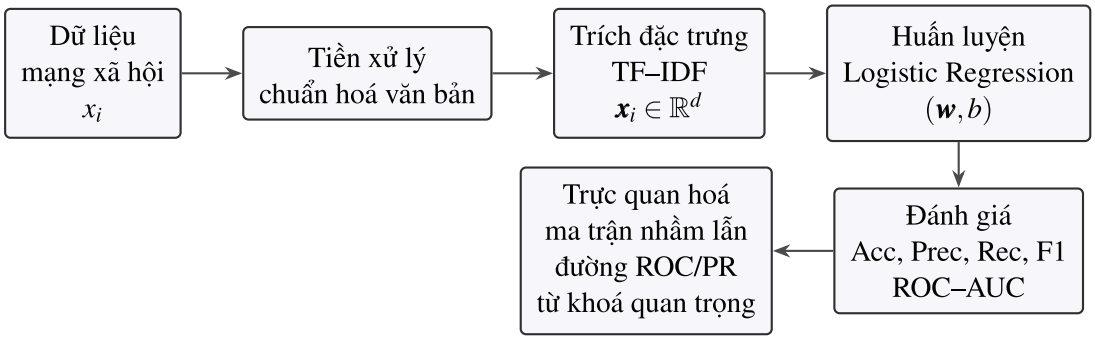
  từ véc-tơ $\boldsymbol{x}_i$ (Bước 3).
- Định nghĩa và sử dụng các độ đo đánh giá mô hình phân loại như Accuracy, Precision, Recall, F1, ROC–AUC, PR–AUC (Bước 4).
- Thực nghiệm trọn vẹn trên Dataset Twitter US Airline Sentiment, trình bày cấu hình huấn luyện, kết quả đánh giá, trực quan hoá và thử áp dụng mô hình vào một tập bình luận mạng xã hội giả định khoảng 20 câu (Bước 5).

2 Quy trình tổng quát

Toàn bộ hệ thống có thể được mô tả bởi pipeline:

$$x \to \text{Tiền xử lý} \to \text{TF–IDF} \to \text{Logistic Regression} \to \text{Đánh giá} \to$$

$$\to \text{Trực quan hoá \& Diễn giải.}$$



Hình 1: Quy trình tổng quát.

3 Tiền xử lý dữ liệu văn bản mạng xã hội

Tiền xử lý nhằm giảm nhiễu, đồng nhất biểu diễn và cải thiện chất lượng đặc trưng trước khi đưa vào TF–IDF và Logistic Regression. Với văn bản mạng xã hội, đặc biệt là tweet tiếng Anh trong Dataset Twitter US Airline Sentiment, các thao tác thường dùng gồm:

- Chuẩn hoá chữ hoa/thường: chuyển toàn bộ về chữ thường (lowercasing).
- Loại bỏ hoặc chuẩn hoá URL, mention, hashtag:

  $\texttt{https://...} \mapsto \langle \text{URL} \rangle, \qquad \texttt{@user} \mapsto \langle \text{USER} \rangle, \qquad \texttt{\#United} \mapsto \texttt{united}.$

- Xử lý emoji/ký hiệu: có thể giữ lại (để mô hình học được tín hiệu cảm xúc) hoặc ánh xạ về token cảm xúc, tuỳ mục tiêu ứng dụng.
- Chuẩn hoá dấu câu và khoảng trắng: loại bỏ lặp ký tự, chuẩn hoá nhiều khoảng trắng liên tiếp.
- Tách từ (tokenization): với tiếng Anh có thể dùng tách theo khoảng trắng kết hợp quy tắc đơn giản để tách dấu câu, với tiếng Việt nên dùng bộ tách từ chuyên biệt.
- Loại bỏ stopwords (tuỳ chọn): chỉ nên làm khi stopwords không mang nhiều thông tin cảm xúc trong ngữ cảnh bài toán.

Ngoài các thao tác trên, với dữ liệu mạng xã hội thường cần xử lý thêm các hiện tượng đặc thù:

- Chuẩn hoá kéo dài ký tự (ví dụ $\texttt{soooo} \mapsto \texttt{so}$) và lặp dấu câu (ví dụ $\texttt{!!!} \mapsto \texttt{!}$).
- Chuẩn hoá viết tắt, tiếng lóng nếu xuất hiện lặp lại nhiều lần trong dữ liệu (ví dụ $\texttt{u} \mapsto \texttt{you}$, $\texttt{pls} \mapsto \texttt{please}$).

- Chuẩn hoá Unicode và loại bỏ ký tự điều khiển để tránh cùng một ký tự có nhiều mã hoá khác nhau.
- Cân nhắc giữ các từ phủ định (not, never, no, cannot) và dấu !/? vì chúng thường mang tín hiệu cảm xúc mạnh.

Trong thực nghiệm, toàn bộ tiền xử lý nên được đóng gói thành một hàm hoặc một bước trong pipeline để có thể áp dụng giống nhau cho cả tập huấn luyện, validation, test và dữ liệu mới. Tuyệt đối không điều chỉnh quy tắc tiền xử lý dựa trên quan sát tập test để tránh rò rỉ thông tin.

(a) Quan sát 10 tweet mẫu trước và sau tiền xử lý

Để hình dung rõ hơn, ta trích ra 10 tweet đầu tiên (hoặc bất kỳ 10 tweet tiêu biểu) từ Dataset Twitter US Airline Sentiment, chỉ giữ lại cột văn bản `text` và nhãn `airline_sentiment`. Bảng dưới đây minh hoạ dạng dữ liệu (đã rút gọn nội dung cho ngắn gọn):

ID	Nhãn gốc	Tweet gốc (rút gọn)	Sau tiền xử lý cơ bản
1	positive	I love flying with United, great service!	i love flying with united great service
2	negative	Flight delayed 3 hours, very unhappy with @AmericanAir	flight delayed 3 hours very unhappy with <user>
3	negative	Worst flight ever. Lost my bag and no help from staff...	worst flight ever lost my bag and no help from staff
4	positive	Amazing crew today on Delta, thank you for the upgrade :)	amazing crew today on delta thank you for the upgrade
5	negative	Cancelled flight again. This is ridiculous!!!	cancelled flight again this is ridiculous
6	positive	On time and smooth flight, really impressed.	on time and smooth flight really impressed
7	negative	Stuck on the runway for 2 hours, no info from the captain	stuck on the runway for 2 hours no info from the captain
8	negative	Customer service line kept me on hold forever...	customer service line kept me on hold forever
9	positive	Free snacks and friendly attendants, great experience	free snacks and friendly attendants great experience
10	negative	Missed my connection because of your delay, not happy	missed my connection because of your delay not happy

Bảng 1: Một số tweet mẫu trước và sau tiền xử lý cơ bản

Trong mã nguồn thực nghiệm, có thể thực hiện bước này bằng cách dùng `pandas` để đọc tệp `CSV`, sau đó in ra vài dòng đầu và một số dòng ngẫu nhiên để kiểm tra nhanh chất lượng dữ liệu và quy tắc tiền xử lý.

(b) Trích một tập con nhỏ để minh hoạ cho các bước tiếp theo

Để việc tính toán bằng tay trở nên khả thi và dễ theo dõi, ta trích một tập con rất nhỏ gồm bốn tweet đã được tiền xử lý, ký hiệu là

$$x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}.$$

Ví dụ:

$$x^{(1)} : \text{great flight no delay friendly service},$$
$$x^{(2)} : \text{terrible delay bad service},$$
$$x^{(3)} : \text{flight delay good service},$$
$$x^{(4)} : \text{good flight good service}.$$

Giả sử nhãn nhị phân tương ứng (sau khi bỏ lớp neutral và ánh xạ positive $\mapsto 1$, negative $\mapsto 0$) là:

$$y^{(1)} = 1, \quad y^{(2)} = 0, \quad y^{(3)} = 1, \quad y^{(4)} = 1.$$

Trong các bước tiếp theo, ta sẽ dùng tập con này để:

- Tự tay xây dựng từ vựng nhỏ, tính toán TF, IDF và TF–IDF cho một vài từ khoá.
- Viết ma trận đặc trưng dạng đại số tuyến tính.
- Minh hoạ cách Logistic Regression nhận đầu vào là các véc-tơ TF–IDF và tạo ra xác suất dự đoán.

4 Biểu diễn văn bản bằng TF–IDF và ngôn ngữ đại số tuyến tính

(a) Từ văn bản sang véc-tơ đặc trưng

Logistic Regression làm việc trên véc-tơ đặc trưng số học, do đó mỗi văn bản x_i cần được ánh xạ thành một véc-tơ

$$\boldsymbol{x}_i = \phi(x_i) \in \mathbb{R}^d,$$

trong đó d là kích thước từ vựng (vocabulary) sau tiền xử lý, và $\phi(\cdot)$ là phép biến đổi TF–IDF.

Ký hiệu $\mathcal{V} = \{t_1, \ldots, t_d\}$ là tập từ vựng thu được sau khi:

- Áp dụng toàn bộ các bước tiền xử lý cho toàn bộ tập huấn luyện.
- Đếm tần suất xuất hiện của các từ trong tập huấn luyện.
- Loại bỏ các từ quá hiếm hoặc quá phổ biến theo các ngưỡng `min_df`, `max_df`.
- Giới hạn số lượng từ giữ lại bởi tham số `max_features` (nếu cần).

Với mỗi văn bản x_i, véc-tơ TF–IDF được viết dưới dạng

$$\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})^\top,$$

trong đó x_{ij} là trọng số TF–IDF của từ t_j trong văn bản x_i.

(b) Định nghĩa TF, IDF và TF–IDF

Cho một tập gồm n văn bản đã tiền xử lý $x^{(1)},\dots,x^{(n)}$ và một từ vựng $\mathcal{V} = \{t_1,\dots,t_d\}$. Khi đó:

- Tần suất xuất hiện của từ t_j trong văn bản $x^{(i)}$ (term frequency) được ký hiệu là
$$\mathrm{tf}_{ij} = \text{số lần từ } t_j \text{ xuất hiện trong văn bản } x^{(i)}.$$
- Số văn bản có chứa từ t_j (document frequency) là
$$\mathrm{df}_j = \left|\{\, i \in \{1,\dots,n\} : t_j \in x^{(i)} \,\}\right|.$$
- Inverse Document Frequency (IDF) đo lường mức độ đặc trưng của một từ trên toàn bộ tập văn bản. Một định nghĩa đơn giản thường dùng là
$$\mathrm{idf}_j = \log \frac{n}{\mathrm{df}_j}.$$
Nếu một từ xuất hiện trong hầu hết các văn bản (df_j lớn), thì idf_j nhỏ; ngược lại, nếu từ xuất hiện trong ít văn bản, idf_j lớn hơn.
- Trọng số TF–IDF của từ t_j trong văn bản $x^{(i)}$ được định nghĩa là
$$x_{ij} = \mathrm{tf}_{ij} \cdot \mathrm{idf}_j.$$

Trong các thư viện thực tế (như `scikit-learn`), định nghĩa IDF có thể được điều chỉnh nhẹ để tăng ổn định số, ví dụ:

$$\mathrm{idf}_j = \log \frac{n+1}{\mathrm{df}_j + 1} + 1,$$

nhưng về mặt trực giác ý nghĩa vẫn giống nhau: TF đo mức độ quan trọng của từ trong từng văn bản, IDF giảm ảnh hưởng của các từ xuất hiện ở mọi nơi, và TF–IDF là sự kết hợp của hai yếu tố đó.

(c) Ví dụ tính TF–IDF trên tập con bốn tweet

Xét lại bốn văn bản đã được tiền xử lý ở bước trên:

$$x^{(1)} : \text{great flight no delay friendly service},$$
$$x^{(2)} : \text{terrible delay bad service},$$
$$x^{(3)} : \text{flight delay good service},$$
$$x^{(4)} : \text{good flight good service}.$$

Để minh hoạ, ta chỉ xét một từ vựng rất nhỏ gồm năm từ khoá liên quan cảm xúc và ngữ cảnh:

$$\mathcal{V}_{\text{toy}} = \{\text{good}, \text{bad}, \text{delay}, \text{service}, \text{flight}\}.$$

Bước 1: Đếm tần suất tf_{ij} của từng từ trong từng văn bản:

Văn bản	good	bad	delay	service	flight
$x^{(1)}$	0	0	1	1	1
$x^{(2)}$	0	1	1	1	0
$x^{(3)}$	1	0	1	1	1
$x^{(4)}$	2	0	0	1	1

Bảng 2: Tần suất xuất hiện tf_{ij} trên tập con bốn tweet

Từ bảng trên, ta có số văn bản chứa mỗi từ (document frequency):

$$\text{df}_{\text{good}} = 2, \quad \text{df}_{\text{bad}} = 1, \quad \text{df}_{\text{delay}} = 3, \quad \text{df}_{\text{service}} = 4, \quad \text{df}_{\text{flight}} = 3.$$

Vì có $n = 4$ văn bản, nên IDF theo công thức $\text{idf}_j = \log \frac{n}{\text{df}_j}$ là:

$$\text{idf}_{\text{good}} \approx \log \frac{4}{2} \approx 0{,}69, \quad \text{idf}_{\text{bad}} \approx \log \frac{4}{1} \approx 1{,}39,$$

$$\text{idf}_{\text{delay}} \approx \log \frac{4}{3} \approx 0{,}29, \quad \text{idf}_{\text{service}} = \log \frac{4}{4} = 0, \quad \text{idf}_{\text{flight}} \approx \log \frac{4}{3} \approx 0{,}29.$$

Ta nhận thấy:

- Từ `service` xuất hiện trong cả bốn văn bản nên $\text{idf}_{\text{service}} = 0$, tức là nó không giúp phân biệt giữa các văn bản trong tập con này.
- Từ `bad` chỉ xuất hiện trong một văn bản duy nhất nên có IDF lớn hơn, phản ánh tính đặc trưng cao.

Bước 2: Tính TF–IDF bằng công thức $x_{ij} = \text{tf}_{ij} \cdot \text{idf}_j$ (làm tròn đến ba chữ số thập phân):

Văn bản	good	bad	delay	service	flight
$x^{(1)}$	0.000	0.000	0.288	0.000	0.288
$x^{(2)}$	0.000	1.386	0.288	0.000	0.000
$x^{(3)}$	0.693	0.000	0.288	0.000	0.288
$x^{(4)}$	1.386	0.000	0.000	0.000	0.288

Bảng 3: Trọng số TF–IDF (làm tròn) trên tập con bốn tweet

Từ bảng TF–IDF, ta có thể rút ra trực giác:

- Văn bản $x^{(2)}$ có trọng số lớn ở từ `bad` và `delay`, gợi ý cảm xúc tiêu cực.
- Các văn bản $x^{(3)}$ và $x^{(4)}$ có trọng số lớn ở từ `good`, gợi ý cảm xúc tích cực.
- Từ `service` xuất hiện ở mọi nơi nên có trọng số bằng 0 và không giúp phân biệt trong ví dụ này.

(d) Ma trận tài liệu–từ vựng và góc nhìn đại số tuyến tính

Nếu xếp các véc-tơ TF–IDF thành hàng, ta thu được ma trận đặc trưng:

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \boldsymbol{x}_2^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d}.$$

Trong ví dụ thu gọn trên với bốn văn bản và năm từ khoá, ma trận đặc trưng rút gọn có dạng:

$$\boldsymbol{X}_{\text{sample}} = \begin{bmatrix} 0 & 0.000 & 0.288 & 0 & 0.288 \\ 0 & 1.386 & 0.288 & 0 & 0 \\ 0.693 & 0 & 0.288 & 0 & 0.288 \\ 1.386 & 0 & 0 & 0 & 0.288 \end{bmatrix}.$$

Trên ma trận này, nhiều thao tác của hệ thống AI có thể được mô tả rõ ràng bằng ngôn ngữ đại số tuyến tính:

- Điểm số tuyến tính của Logistic Regression cho từng văn bản là

  $$z_i = \boldsymbol{w}^\top \boldsymbol{x}_i + b,$$

  tức là tích vô hướng giữa véc-tơ trọng số $\boldsymbol{w}$ và véc-tơ đặc trưng $\boldsymbol{x}_i$ cộng với hệ số chệch b.

- Độ tương đồng cosine giữa hai văn bản (hữu ích cho việc kiểm tra dữ liệu hoặc gợi ý các văn bản gần nhau) là

  $$\cos(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i^\top \boldsymbol{x}_j}{\|\boldsymbol{x}_i\|_2 \|\boldsymbol{x}_j\|_2}.$$

- Giảm chiều và trực quan hoá (tuỳ chọn): có thể dùng SVD hoặc PCA để chiếu các hàng của $\boldsymbol{X}$ xuống $\mathbb{R}^2$ hoặc $\mathbb{R}^3$ để quan sát phân bố dữ liệu theo cảm xúc.

Như vậy, sau bước trên, toàn bộ tập dữ liệu văn bản đã được chuyển thành một ma trận số học, trên đó Logistic Regression và các phương pháp đại số tuyến tính khác có thể được áp dụng một cách tự nhiên.

5 Logistic Regression: mô hình, hàm mất mát và tối ưu

(a) Mô hình xác suất và hàm sigmoid

Với mỗi văn bản x_i sau khi đã được ánh xạ thành véc-tơ TF–IDF $\boldsymbol{x}_i \in \mathbb{R}^d$, Logistic Regression mô hình hoá trực tiếp xác suất để văn bản đó thuộc lớp tích cực:

$$\hat{p}_i = p(y_i = 1 \mid \boldsymbol{x}_i) = \sigma(z_i), \qquad z_i = \boldsymbol{w}^\top \boldsymbol{x}_i + b,$$

trong đó $\boldsymbol{w} \in \mathbb{R}^d$ là véc-tơ trọng số, $b \in \mathbb{R}$ là hệ số chệch, và $\sigma(\cdot)$ là hàm sigmoid:

$$\sigma(z) = \frac{1}{1+\exp(-z)}.$$

Như vậy, trên không gian đặc trưng, Logistic Regression là một mô hình tuyến tính theo $\boldsymbol{x}_i$ thông qua điểm số

$$z_i = \boldsymbol{w}^\top \boldsymbol{x}_i + b,$$

sau đó dùng một phép biến đổi phi tuyến (sigmoid) để ánh xạ $z_i \in \mathbb{R}$ về khoảng $(0,1)$, phù hợp với ý nghĩa xác suất.

Một số tính chất quan trọng của hàm sigmoid:

- $\sigma(z) \in (0,1)$ với mọi $z \in \mathbb{R}$, nên có thể diễn giải như một xác suất.
- $\sigma(0) = \dfrac{1}{2}$. Điểm $z = 0$ là ngưỡng cân bằng, mô hình phân vân giữa hai lớp.
- Khi $z \to +\infty$ thì $\sigma(z) \to 1$, còn khi $z \to -\infty$ thì $\sigma(z) \to 0$. Điểm số z_i càng dương thì mô hình càng tin rằng $y_i = 1$, càng âm thì càng tin $y_i = 0$.

Nếu chọn ngưỡng quyết định $\tau = 0.5$, quy tắc phân loại trở thành

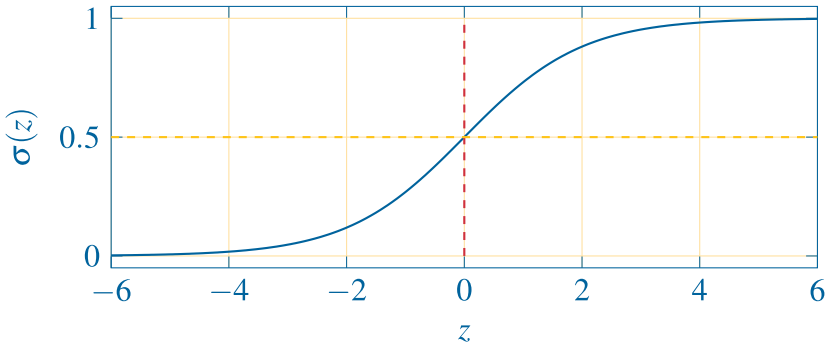
$$\hat{y}_i = \begin{cases} 1, & \hat{p}_i \geq 0.5, \\ 0, & \hat{p}_i < 0.5. \end{cases}$$

Vì $\sigma(z) \geq 0.5$ khi và chỉ khi $z \geq 0$, nên quy tắc trên tương đương với

$$\hat{y}_i = \begin{cases} 1, & \boldsymbol{w}^\top \boldsymbol{x}_i + b \geq 0, \\ 0, & \boldsymbol{w}^\top \boldsymbol{x}_i + b < 0. \end{cases}$$

Do đó, với ngưỡng chuẩn 0.5, Logistic Regression chính là một bộ phân loại tuyến tính với siêu phẳng quyết định

$$\boldsymbol{w}^\top \boldsymbol{x} + b = 0.$$



Hình 2: Hàm sigmoid ánh xạ điểm số tuyến tính z về xác suất $\sigma(z)$

(b) Diễn giải theo odds và log-odds

Để hiểu rõ hơn ý nghĩa của các trọng số trong Logistic Regression, ta xét tỉ số odds (tỉ số có–không) và log-odds. Với một mẫu bất kỳ:

$$\text{odds}_i = \frac{\hat{p}_i}{1-\hat{p}_i}, \qquad \log \frac{\hat{p}_i}{1-\hat{p}_i} = \boldsymbol{w}^\top \boldsymbol{x}_i + b.$$

Nói cách khác, Logistic Regression giả định rằng log-odds của sự kiện $y_i = 1$ là một hàm tuyến tính của véc-tơ đặc trưng $\boldsymbol{x}_i$. Điều này làm cho mô hình có tính diễn giải cao:

- Nếu $w_j > 0$, khi thành phần x_{ij} tăng lên (giữ các thành phần khác cố định), log-odds tăng, odds tăng và xác suất $\hat{p}_i$ có xu hướng tăng.
- Nếu $w_j < 0$, khi x_{ij} tăng, log-odds giảm, odds giảm và xác suất $\hat{p}_i$ có xu hướng giảm.

Cụ thể hơn, nếu x_{ij} tăng thêm một đơn vị thì log-odds tăng thêm w_j, còn odds được nhân với $\exp(w_j)$. Ví dụ, nếu $w_j \approx 0{,}7$ thì $\exp(w_j) \approx 2$, nghĩa là odds tăng gần gấp đôi.

(c) Hàm mất mát cross-entropy và regularization

Đặt giả thiết xác suất:

$$y_i \mid \boldsymbol{x}_i \sim \text{Bernoulli}(\hat{p}_i), \qquad \hat{p}_i = p(y_i = 1 \mid \boldsymbol{x}_i).$$

Khi đó, likelihood của toàn bộ tập dữ liệu là

$$\mathscr{P}(\boldsymbol{y} \mid \boldsymbol{X}; \boldsymbol{w}, b) = \prod_{i=1}^{n} \hat{p}_i^{\,y_i} (1-\hat{p}_i)^{1-y_i}.$$

Trong đó:

- $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ là véc-tơ nhãn quan sát, với $y_i \in \{0, 1\}$.
- $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ là ma trận đặc trưng, hàng thứ i là $\boldsymbol{x}_i^\top$.
- $\boldsymbol{w} \in \mathbb{R}^d$, $b \in \mathbb{R}$ là tham số của Logistic Regression.
- $\hat{p}_i = p(y_i = 1 \mid \boldsymbol{x}_i; \boldsymbol{w}, b) = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b)$ là xác suất mô hình gán cho việc mẫu thứ i thuộc lớp 1.

Giả thiết $y_i \mid \boldsymbol{x}_i \sim \text{Bernoulli}(\hat{p}_i)$ và các mẫu độc lập có điều kiện theo $\boldsymbol{X}$ cho ta:

$$\mathscr{P}(\boldsymbol{y} \mid \boldsymbol{X}; \boldsymbol{w}, b) = \prod_{i=1}^{n} \hat{p}_i^{\,y_i} (1 - \hat{p}_i)^{1 - y_i}.$$

Mỗi thừa số

$$\hat{p}_i^{\,y_i} (1 - \hat{p}_i)^{1 - y_i} = \begin{cases} \hat{p}_i, & \text{nếu } y_i = 1, \\ 1 - \hat{p}_i, & \text{nếu } y_i = 0, \end{cases}$$

chính là xác suất để mô hình sinh ra nhãn y_i tại mẫu thứ i. Tích của các thừa số này là xác suất (likelihood) để mô hình với tham số $(\boldsymbol{w}, b)$ sinh ra đúng toàn bộ véc-tơ nhãn quan sát $\boldsymbol{y}$ khi dữ liệu đầu vào là $\boldsymbol{X}$.

Ví dụ, với $n = 3$ và các nhãn quan sát

$$(y_1, y_2, y_3) = (1, 0, 1),$$

ta có

$$\mathscr{P}(\boldsymbol{y} \mid \boldsymbol{X}; \boldsymbol{w}, b) = \hat{p}_1^{\,1} (1 - \hat{p}_1)^0 \cdot \hat{p}_2^{\,0} (1 - \hat{p}_2)^1 \cdot \hat{p}_3^{\,1} (1 - \hat{p}_3)^0 = \hat{p}_1 (1 - \hat{p}_2) \hat{p}_3.$$

Lấy log, ta được log-likelihood:

$$\ell(\boldsymbol{w}, b) = \sum_{i=1}^{n} \Big(y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i) \Big).$$

Tối đa hoá log-likelihood tương đương với tối thiểu hoá hàm mất mát là trung bình âm log-likelihood, thường gọi là hàm mất mát cross-entropy:

$$\mathscr{L}(\boldsymbol{w}, b) = -\frac{1}{n} \sum_{i=1}^{n} \Big(y_i \log \hat{p}_i + (1 - y_i) \log(1 - \hat{p}_i) \Big).$$

Từng hạng tử trong tổng có thể hiểu như sau: nếu nhãn thật là $y_i = 1$ mà mô hình cho $\hat{p}_i$ nhỏ, thì $-\log \hat{p}_i$ rất lớn, mô hình bị phạt nặng; ngược lại nếu $y_i = 0$ mà $\hat{p}_i$ lại lớn thì $-\log(1 - \hat{p}_i)$ rất lớn.

Để hạn chế hiện tượng overfitting, tức là khi mô hình rất sát dữ liệu huấn luyện nhưng dự đoán kém trên dữ liệu mới, người ta thường bổ sung một hạng tử điều

chuẩn (regularization) nhằm phạt các nghiệm có véc-tơ trọng số có chuẩn quá lớn. Với regularization ℓ_2, ta xét hàm mục tiêu:

$$\mathscr{J}(\boldsymbol{w}, b) = \mathscr{L}(\boldsymbol{w}, b) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2,$$

trong đó $\lambda > 0$ điều khiển mức độ phạt. Khi λ lớn, các trọng số bị kéo về gần 0 mạnh hơn, mô hình ít phức tạp hơn nhưng có thể tăng bias; khi λ nhỏ, mô hình linh hoạt hơn nhưng dễ bị overfitting.

(d) Gradient và quy tắc cập nhật cơ bản

Nhắc lại, ta đã định nghĩa hàm mục tiêu (với điều chuẩn ℓ_2) là

$$\mathscr{J}(\boldsymbol{w}, b) = \mathscr{L}(\boldsymbol{w}, b) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2,$$

trong đó $\mathscr{L}$ là hàm mất mát cross-entropy và $\lambda > 0$ là hệ số điều chuẩn.

Khái niệm gradient có thể xem như phần mở rộng của đạo hàm sang trường hợp nhiều biến. Với một hàm vô hướng $f(\boldsymbol{w})$ phụ thuộc vào véc-tơ $\boldsymbol{w} \in \mathbb{R}^d$, gradient của f theo $\boldsymbol{w}$ là véc-tơ

$$\nabla_{\boldsymbol{w}} f(\boldsymbol{w}) = \left(\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \ldots, \frac{\partial f}{\partial w_d} \right)^\top.$$

Mỗi thành phần là một đạo hàm riêng theo từng biến. Gradient chỉ ra hướng tăng nhanh nhất của hàm; ngược lại, hướng $-\nabla_{\boldsymbol{w}} f$ là hướng giảm mạnh nhất.

Trong bài toán Logistic Regression, ta cần gradient của $\mathscr{J}$ theo $\boldsymbol{w}$ và đạo hàm theo b. Ký hiệu $\hat{\boldsymbol{p}} = (\hat{p}_1, \ldots, \hat{p}_n)^\top$ là véc-tơ xác suất dự đoán và $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ là véc-tơ nhãn quan sát, ta có (với regularization ℓ_2):

$$\nabla_{\boldsymbol{w}} \mathscr{J} = \frac{1}{n} \boldsymbol{X}^\top (\hat{\boldsymbol{p}} - \boldsymbol{y}) + \lambda \boldsymbol{w}, \qquad \frac{\partial \mathscr{J}}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} (\hat{p}_i - y_i).$$

Trong biểu thức trên:

- $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ là ma trận đặc trưng, hàng thứ i là $\boldsymbol{x}_i^\top$.
- $\hat{p}_i = \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i + b)$ là xác suất mô hình gán cho việc $y_i = 1$.
- Hiệu $\hat{p}_i - y_i$ chính là sai số dự đoán trên mẫu thứ i ở dạng xác suất.

Có thể hiểu như sau: gradient theo $\boldsymbol{w}$ là trung bình có trọng số của các véc-tơ đặc trưng $\boldsymbol{x}_i$, với trọng số chính là sai số $(\hat{p}_i - y_i)$, cộng thêm hạng tử $\lambda \boldsymbol{w}$ do điều chuẩn. Nếu mô hình dự đoán tốt trên một mẫu (ví dụ $y_i = 1$ và $\hat{p}_i$ gần 1) thì $\hat{p}_i - y_i$ nhỏ, đóng góp của mẫu đó vào gradient nhỏ. Nếu mô hình dự đoán kém (ví dụ $y_i = 0$

nhưng $\hat{p}_i$ lại lớn) thì $\hat{p}_i - y_i$ có độ lớn đáng kể, mẫu đó sẽ kéo gradient mạnh hơn và tác động nhiều hơn lên bước cập nhật tham số.

Ở trường hợp một biến, nếu ta có một hàm $g(w)$ và đạo hàm $g'(w)$ dương, điều đó cho thấy khi w tăng thì $g(w)$ tăng; muốn giảm giá trị hàm, ta phải dịch w theo hướng ngược lại, tức là $w \leftarrow w - \eta g'(w)$ với $\eta > 0$ nhỏ. Gradient Descent tổng quát hoá ý tưởng này cho véc-tơ nhiều chiều.

Với Gradient Descent, tham số được cập nhật theo bước học (learning rate) $\eta > 0$:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla_{\boldsymbol{w}} \mathscr{J}, \qquad b \leftarrow b - \eta \frac{\partial \mathscr{J}}{\partial b}.$$

Nói cách khác, ta di chuyển cặp $(\boldsymbol{w}, b)$ theo hướng ngược chiều gradient để làm giảm giá trị của hàm mục tiêu $\mathscr{J}$.

Trong thực nghiệm, tiêu chuẩn dừng có thể dựa trên số vòng lặp tối đa, sự thay đổi của hàm mục tiêu $\mathscr{J}$ giữa hai vòng lặp liên tiếp, hoặc chuẩn của gradient. Do hàm mục tiêu của Logistic Regression với điều chuẩn ℓ_2 là hàm lồi theo $(\boldsymbol{w}, b)$, mọi điểm cực tiểu địa phương cũng là cực tiểu toàn cục. Nhờ đó, các thuật toán tối ưu như Gradient Descent, mini-batch hoặc Stochastic Gradient Descent, hay các phương pháp bậc hai (L-BFGS, Newton) đều có thể hội tụ về nghiệm tối ưu toàn cục nếu chọn siêu tham số phù hợp.

(e) Xét Logistic Regression trên tập con bốn tweet

Quay lại ma trận TF–IDF rút gọn ở Bước 2 với bốn văn bản và năm từ khoá, ta giả sử mô hình Logistic Regression chỉ sử dụng hai đặc trưng quan trọng nhất là trọng số của các từ `good` và `bad`. Khi đó, véc-tơ đặc trưng rút gọn tương ứng cho bốn văn bản là:

$$\tilde{\boldsymbol{x}}^{(1)} = (0,\, 0), \quad \tilde{\boldsymbol{x}}^{(2)} = (0,\, 1{,}386), \quad \tilde{\boldsymbol{x}}^{(3)} = (0{,}693,\, 0), \quad \tilde{\boldsymbol{x}}^{(4)} = (1{,}386,\, 0).$$

Giả sử ta chọn một véc-tơ trọng số minh hoạ

$$\boldsymbol{w} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad b = 0.$$

Thành phần thứ nhất (ứng với từ `good`) có trọng số dương, thành phần thứ hai (ứng với từ `bad`) có trọng số âm, phù hợp với trực giác: từ good làm tăng xác suất tích cực, từ bad làm giảm xác suất tích cực.

Ta tính điểm số và xác suất cho từng văn bản:

$$z^{(i)} = \boldsymbol{w}^{\top} \tilde{\boldsymbol{x}}^{(i)} + b, \qquad \hat{p}^{(i)} = \sigma\big(z^{(i)}\big).$$

Cụ thể:

- Văn bản $x^{(2)}$ có nhiều nội dung tiêu cực (từ `bad`):

$$z^{(2)} = 1 \cdot 0 + (-1) \cdot 1{,}386 \approx -1{,}386$$

nên

$$\hat{p}^{(2)} = \sigma(-1{,}386) \approx \frac{1}{1+\exp(1{,}386)} \approx 0{,}2.$$

Xác suất tích cực khoảng 20%, mô hình có xu hướng dự đoán đây là tweet tiêu cực ($\hat{y}^{(2)} = 0$).

- Văn bản $x^{(4)}$ chứa nhiều từ `good`:

$$z^{(4)} = 1 \cdot 1{,}386 + (-1) \cdot 0 = 1{,}386$$

nên

$$\hat{p}^{(4)} = \sigma(1{,}386) \approx \frac{1}{1+\exp(-1{,}386)} \approx 0{,}8.$$

Xác suất tích cực khoảng 80%, mô hình có xu hướng dự đoán đây là tweet tích cực ($\hat{y}^{(4)} = 1$).

Ví dụ mẫu này cho thấy cách TF–IDF và Logistic Regression kết hợp với nhau:

- TF–IDF chuyển mỗi văn bản thành véc-tơ số học, trong đó các từ mang tính phân biệt (good, bad, delay, ...) có trọng số khác nhau.
- Logistic Regression học một véc-tơ trọng số $\boldsymbol{w}$ để ánh xạ các véc-tơ đó sang xác suất thuộc lớp tích cực, với ý nghĩa rõ ràng theo log-odds.

Trong phần thực nghiệm (Bước 5), ta sẽ thay tập con nhỏ này bằng toàn bộ Dataset Twitter US Airline Sentiment đã tiền xử lý và tối ưu hoá các trọng số $(\boldsymbol{w}, b)$ bằng các thư viện học máy như `scikit-learn`.

6 Độ đo đánh giá mô hình phân loại

(a) Ma trận nhầm lẫn

Ma trận nhầm lẫn (confusion matrix) là bảng tóm tắt cách mô hình phân loại gán nhãn so với nhãn thật trên một tập dữ liệu kiểm tra. Với bài toán phân loại nhị phân, ta quy ước:

$$y \in \{0,1\}, \qquad \hat{y} \in \{0,1\},$$

trong đó y là nhãn thật, $\hat{y}$ là nhãn mô hình dự đoán. Ma trận nhầm lẫn được viết dưới dạng:

	$\hat{y}=1$	$\hat{y}=0$
$y=1$	TP	FN
$y=0$	FP	TN

Trong đó mỗi ô là số lượng mẫu rơi vào trường hợp tương ứng:

- TP (true positive): số mẫu có nhãn thật $y = 1$ và mô hình dự đoán đúng $\hat{y} = 1$ (đúng tích cực).
- TN (true negative): số mẫu có nhãn thật $y = 0$ và mô hình dự đoán đúng $\hat{y} = 0$ (đúng tiêu cực).
- FP (false positive): số mẫu có nhãn thật $y = 0$ nhưng mô hình dự đoán nhầm thành $\hat{y} = 1$ (sai tích cực).
- FN (false negative): số mẫu có nhãn thật $y = 1$ nhưng mô hình dự đoán nhầm thành $\hat{y} = 0$ (sai tiêu cực).

Có thể hiểu: mỗi hàng của ma trận tương ứng với nhãn thật, mỗi cột tương ứng với nhãn dự đoán. Đường chéo TP và TN thể hiện số lần mô hình dự đoán đúng, hai ô FP và FN thể hiện số lần mô hình dự đoán sai theo hai kiểu khác nhau.

Ví dụ, nếu trên một tập kiểm tra có 100 mẫu, ta thu được

$$\mathrm{TP} = 50, \quad \mathrm{FN} = 10, \quad \mathrm{FP} = 5, \quad \mathrm{TN} = 35,$$

thì mô hình đã dự đoán đúng 85 mẫu (50 tích cực và 35 tiêu cực), dự đoán sai 15 mẫu (10 trường hợp lẽ ra tích cực nhưng bị đoán thành tiêu cực, 5 trường hợp lẽ ra tiêu cực nhưng bị đoán thành tích cực). Các độ đo như Accuracy, Precision, Recall, F1 ở các mục sau đều được xác định trực tiếp từ bốn số lượng TP, TN, FP, FN này.

(b) Accuracy, Precision, Recall, F1

Với bài toán nhị phân, kết quả dự đoán có thể tóm tắt bằng bốn đại lượng: TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative). Từ đó, các độ đo cơ bản được định nghĩa như sau:

$$\mathrm{Accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}},$$

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \qquad \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \qquad \mathrm{F1} = \frac{2\,\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}.$$

Accuracy đo tỉ lệ dự đoán đúng trên toàn bộ mẫu. Precision cho biết trong số các mẫu được mô hình gán vào lớp dương, có bao nhiêu phần trăm thực sự là dương. Recall cho biết trong số các mẫu dương thật, mô hình đã phát hiện được bao nhiêu phần trăm. F1 là trung bình điều hoà giữa Precision và Recall, chỉ cao khi cả hai đồng thời cao.

Để trực quan, xét ví dụ nhỏ với 100 mẫu, trong đó 40 mẫu dương thật, 60 mẫu âm thật. Giả sử mô hình dự đoán:

$$\mathrm{TP} = 30, \quad \mathrm{FN} = 10, \quad \mathrm{FP} = 5, \quad \mathrm{TN} = 55.$$

Khi đó:

$$\text{Accuracy} = \frac{30+55}{100} = 0{,}85, \quad \text{Precision} = \frac{30}{30+5} = 0{,}857, \quad \text{Recall} = \frac{30}{30+10} = 0{,}75,$$

$$\text{F1} = \frac{2 \cdot 0{,}857 \cdot 0{,}75}{0{,}857+0{,}75} \approx 0{,}799.$$

Ta thấy Accuracy khá cao, nhưng vẫn còn 25% mẫu dương bị bỏ sót (Recall = 0,75), điều này có thể quan trọng trong các bài toán nhạy cảm như phát hiện khiếu nại, cảnh báo rủi ro.

Trong trường hợp dữ liệu mất cân bằng (một lớp chiếm đa số), Accuracy dễ gây ảo giác. Ví dụ, nếu 95% tweet là tiêu cực và mô hình luôn dự đoán tiêu cực, Accuracy đạt 95%, nhưng Recall của lớp tích cực bằng 0. Khi đó nên ưu tiên phân tích Precision, Recall, F1 hoặc các độ đo dựa trên đường cong Precision–Recall (PR–AUC).

Một số độ đo bổ sung thường dùng:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}}, \qquad \text{Balanced Accuracy} = \frac{1}{2}\Big(\text{Recall} + \text{Specificity}\Big).$$

Specificity là tỉ lệ mẫu âm thật được dự đoán đúng là âm (còn gọi là Recall của lớp âm). Balanced Accuracy lấy trung bình Recall của hai lớp, do đó phản ánh công bằng hơn trong trường hợp mất cân bằng lớp.

Nếu bài toán nhiều lớp hoặc cần so sánh công bằng giữa các lớp, thường dùng trung bình macro và micro cho Precision, Recall, F1:

- Macro: tính độ đo riêng cho từng lớp (coi mỗi lớp lần lượt là lớp dương), sau đó lấy trung bình số học. Cách này nhấn mạnh sự đồng đều giữa các lớp, lớp hiếm và lớp phổ biến đều được tính trọng số như nhau.
- Micro: gộp TP, FP, FN của tất cả các lớp rồi tính một độ đo duy nhất. Cách này phản ánh hiệu năng tổng thể có tính đến tần suất xuất hiện của từng lớp, phù hợp khi quan tâm đến độ chính xác toàn cục.

Trong các mô hình như Logistic Regression, đầu ra thường là xác suất $\hat{p} = p(y = 1 \mid \boldsymbol{x})$, sau đó được so sánh với một ngưỡng quyết định τ để suy ra nhãn dự đoán. Ngưỡng τ không nhất thiết phải bằng 0,5. Có thể chọn τ theo mục tiêu vận hành, chẳng hạn:

- tối đa hoá F1,
- ưu tiên Recall với ràng buộc Precision tối thiểu,
- hoặc tối thiểu hoá một hàm chi phí sai phân loại cụ thể.

Khi báo cáo thực nghiệm, cần nêu rõ cách chọn ngưỡng và bảo đảm ngưỡng được hiệu chỉnh trên tập validation riêng (hoặc trong cross-validation), không trực tiếp dùng tập test để điều chỉnh ngưỡng nhằm tránh đánh giá lạc quan quá mức.

(c) ROC, PR và AUC

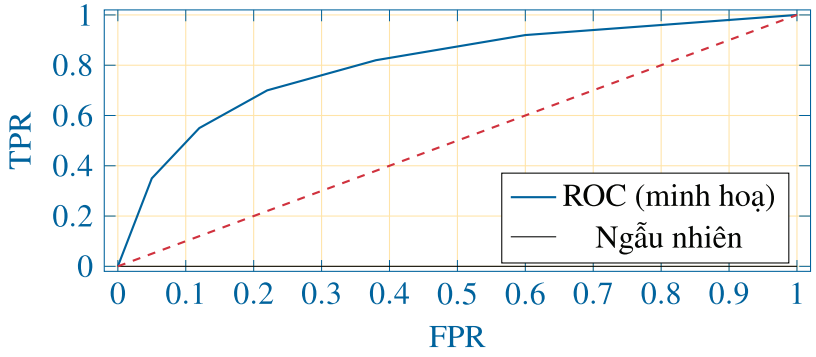
Khi mô hình Logistic Regression cho mỗi mẫu một điểm số hoặc xác suất $\hat{p}_i = p(y_i = 1 \mid \boldsymbol{x}_i)$, việc chọn ngưỡng quyết định τ sẽ dẫn đến các giá trị TP, FP, TN, FN khác nhau. Tương ứng, các độ đo như TPR (True Positive Rate) và FPR (False Positive Rate) thay đổi theo τ:

$$\mathrm{TPR} = \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \qquad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}}.$$

Đường cong ROC (Receiver Operating Characteristic) được xây dựng bằng cách quét ngưỡng τ từ 1 xuống 0, tại mỗi giá trị ngưỡng tính một cặp $(\mathrm{FPR}, \mathrm{TPR})$ và đặt lên mặt phẳng toạ độ. Mỗi điểm trên đường cong ROC tương ứng với một lựa chọn ngưỡng:

- Ngưỡng rất cao (gần 1): mô hình hầu như không dự đoán mẫu dương, TPR và FPR đều gần 0.
- Ngưỡng rất thấp (gần 0): mô hình gần như luôn dự đoán dương, TPR và FPR đều tiến tới 1.
- Ngưỡng trung gian: cân bằng giữa việc phát hiện đúng lớp dương (TPR cao) và hạn chế báo nhầm dương (FPR thấp).

Đường chéo từ $(0,0)$ đến $(1,1)$ biểu diễn bộ phân loại ngẫu nhiên. Một mô hình tốt cần có đường ROC nằm rõ ràng phía trên đường chéo này.



Hình 3: Đường cong ROC (minh hoạ) và đường chuẩn ngẫu nhiên.

Để trực quan, xét bốn mẫu với nhãn thật và xác suất dự đoán:

Mẫu	1	2	3	4
y	1	1	0	0
$\hat{p}$	0.9	0.6	0.4	0.1

Chọn ngưỡng $\tau = 0.5$:

$$\hat{y} = (\mathbf{1}[\hat{p}_i \geq 0.5]) = (1,1,0,0),$$

ta có TP $= 2$, TN $= 2$, FP $= 0$, FN $= 0$, nên

$$\text{TPR} = 1, \qquad \text{FPR} = 0.$$

Nếu giảm ngưỡng xuống $\tau = 0.3$, mọi mẫu đều được dự đoán dương, khi đó TP $= 2$, TN $= 0$, FP $= 2$, FN $= 0$, nên

$$\text{TPR} = 1, \qquad \text{FPR} = 1.$$

Hai điểm $(0,1)$ và $(1,1)$ chính là hai điểm được đặt trên đường ROC khi quét ngưỡng.

Bên cạnh ROC, đường cong Precision–Recall (PR) biểu diễn Precision theo Recall khi thay đổi τ. Khi dữ liệu mất cân bằng, đường PR thường phản ánh rõ hơn chất lượng mô hình trên lớp hiếm, vì Precision giảm mạnh nếu mô hình tạo ra nhiều dương giả (FP), dù Accuracy tổng thể vẫn cao.

Diện tích dưới đường cong (Area Under the Curve, AUC) là một độ đo tổng hợp:

- ROC–AUC nằm trong khoảng $[0{,}5, 1]$ đối với bộ phân loại hợp lý; giá trị 0,5 tương ứng với bộ phân loại ngẫu nhiên, giá trị gần 1 cho thấy mô hình có khả năng phân biệt rất tốt giữa hai lớp trên toàn bộ các ngưỡng.
- PR–AUC có giá trị nền tảng xấp xỉ tỉ lệ mẫu dương trong dữ liệu; càng cao so với mức nền tảng này thì mô hình càng tốt trong việc phát hiện và xác định đúng lớp dương.

Ưu điểm của AUC là không phụ thuộc vào một ngưỡng cụ thể, cho phép so sánh các mô hình về khả năng phân biệt tổng thể trước khi lựa chọn ngưỡng phù hợp với mục tiêu vận hành của hệ thống.

7 Thực nghiệm, huấn luyện mô hình, đánh giá và trực quan hoá

(a) Sử dụng Dataset mở, cấu hình thực nghiệm và đánh giá

Trong phần thực nghiệm này, chúng ta sử dụng một Dataset mở là Twitter US Airline Sentiment (khoảng 14 640 tweets), trong đó mỗi tweet được gán một nhãn cảm xúc thuộc ba lớp: `positive`, `neutral`, `negative`. Với bài toán nhị phân như mô tả ở phần đề bài, ta loại bỏ lớp `neutral` và ánh xạ `positive`$\mapsto 1$, `negative`$\mapsto 0$.

Dữ liệu được cấu trúc và mô tả như sau:

- Định dạng tệp: `CSV` (mỗi dòng là một tweet), có cột văn bản (ví dụ `text`) và cột nhãn (ví dụ `airline_sentiment`).
- Làm sạch dữ liệu: loại bỏ dòng rỗng, loại bỏ trùng lặp (tuỳ chọn), chuẩn hoá Unicode.

- Thống kê dữ liệu: số mẫu theo từng lớp sau khi lọc `neutral`, dùng để nhận diện mức độ mất cân bằng.

Cấu hình thực nghiệm khuyến nghị:

- Ngôn ngữ lập trình: Python (phiên bản ≥ 3.9).
- Thư viện: `numpy`, `pandas`, `scikit-learn`, `matplotlib`, `joblib`.
- Phần cứng: CPU là đủ cho TF–IDF + Logistic Regression; RAM khuyến nghị $\geq$ 8GB nếu `max_features` lớn (có thể sử dụng `https://colab.research.google.com`).
- Tái lập: cố định `random_state`; ghi rõ cấu hình `ngram_range`, `min_df`, `max_df`, `max_features`, `C`, `solver`, `max_iter`.

Phương thức đánh giá đề xuất:

- Tách dữ liệu theo tỉ lệ train/validation/test xấp xỉ $80/10/10$, đồng thời bảo toàn tỉ lệ lớp (sử dụng *stratify*).
- Chỉ sử dụng tập validation để chọn siêu tham số và ngưỡng dự đoán (nếu cần); tập test chỉ dùng để báo cáo cuối cùng, không tham gia hiệu chỉnh mô hình.

(b) Huấn luyện mô hình và lựa chọn siêu tham số

Mô hình được huấn luyện theo pipeline:

$$\text{Văn bản} \to \text{Tiền xử lý} \to \text{TF–IDF} \to \text{Logistic Regression}.$$

Các lựa chọn cấu hình TF–IDF ảnh hưởng trực tiếp đến độ thưa của ma trận $\boldsymbol{X}$ và khả năng tổng quát hoá:

- `ngram_range=(1,2)`: sử dụng unigram và bigram để mô hình nhận diện các cụm từ ngắn.
- `sublinear_tf=True`: dùng $\log(1+\text{tf})$ nhằm giảm tác động của các từ lặp quá nhiều.
- `min_df`: loại bỏ từ quá hiếm (giảm nhiễu); `max_df`: loại bỏ từ quá phổ biến (giảm các từ ít mang tính phân biệt).
- `max_features`: giới hạn kích thước từ vựng để kiểm soát tài nguyên.

Với Logistic Regression, các tham số quan trọng:

- `penalty="l2"`: điều chuẩn ℓ_2 giúp ổn định nghiệm và tăng khả năng tổng quát hoá.
- `C`: nghịch đảo cường độ điều chuẩn; `C` nhỏ tương ứng với mức phạt mạnh hơn, mô hình đơn giản hơn.

- `class_weight="balanced"`: hữu ích khi dữ liệu mất cân bằng sau khi lọc lớp `neutral`.
- `solver="lbfgs"`: phù hợp cho bài toán lồi, hội tụ nhanh; cần tăng `max_iter` nếu cảnh báo chưa hội tụ.

Trong thực nghiệm, siêu tham số nên được lựa chọn trên tập validation (hoặc bằng cross-validation) với một lưới nhỏ, ví dụ:

$$C \in \{0.25, 0.5, 1.0, 2.0, 4.0\}, \qquad \texttt{max_features} \in \{20000, 50000\}.$$

Không điều chỉnh các lựa chọn trên dựa vào kết quả tập test để tránh làm lệch đánh giá.

(c) Đánh giá, báo cáo kết quả và trực quan hoá

Sau huấn luyện, cần báo cáo tối thiểu các chỉ số trên tập test:

- *Classification report*: Precision, Recall, F1 cho từng lớp và các giá trị trung bình (macro, micro, weighted).
- Ma trận nhầm lẫn (Confusion Matrix): cho phép quan sát trực tiếp các loại lỗi FP/FN.
- ROC–AUC và PR–AUC: đánh giá khả năng phân tách của mô hình trên toàn bộ miền ngưỡng; với dữ liệu mất cân bằng, PR–AUC thường phản ánh sát hơn chất lượng mô hình trên lớp hiếm.

Về trực quan hoá, nên tạo và lưu các hình:

- `confusion_matrix.png`: ma trận nhầm lẫn trên tập test.
- `roc_curve.png`: đường ROC và đường chuẩn ngẫu nhiên.
- `pr_curve.png`: đường Precision–Recall và diện tích trung bình (Average Precision).

Để diễn giải mô hình Logistic Regression kết hợp TF–IDF, có thể trích:

- Nhóm từ/cụm từ có trọng số dương lớn nhất: xu hướng góp phần vào quyết định tích cực.
- Nhóm từ/cụm từ có trọng số âm lớn nhất: xu hướng góp phần vào quyết định tiêu cực.

Danh sách này nên được xuất ra tệp `CSV` để phục vụ kiểm tra định tính và thảo luận.

(d) Mã Python thực nghiệm đầy đủ và triển khai trên Colab (kèm 20 bình luận giả định)

Mã dưới đây hiện thực trọn vẹn quy trình:

- Đọc dataset CSV (từ tệp cục bộ hoặc tải về), nếu không có sẽ sinh một dataset giả lập nhỏ để kiểm tra mã nguồn.
- Lọc nhãn để tạo bài toán nhị phân; tiền xử lý văn bản mạng xã hội.
- Chia dữ liệu thành *train/validation/test*.
- Xây dựng pipeline TF–IDF + Logistic Regression, huấn luyện trên tập train.
- Đánh giá, trực quan hoá ROC/PR và ma trận nhầm lẫn, trích các đặc trưng quan trọng, lưu mô hình và kết quả.
- Áp dụng mô hình đã huấn luyện cho 20 bình luận mạng xã hội giả định và in ra nhãn dự đoán cùng xác suất, giúp kiểm tra trực giác.

Mã có thể chạy trên máy cá nhân hoặc trên Google Colab (chỉ cần tải tệp CSV lên hoặc cho phép tải từ URL). Bạn đọc có thể quét QR Code để truy cập chương trình nguồn python trên Google Colab.

Hình 4: QR Code truy cập mã nguồn tại Google Colab

```
# -*- coding: utf-8 -*-
"""
SENTIMENT ANALYSIS SYSTEM (FULL PIPELINE)
Model: TF-IDF + Logistic Regression
Dataset: Twitter US Airline Sentiment (Real Data) + 20
   Mock Sentences (Validation)
Output: English (Best for LaTeX Reporting)
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
import requests
import io
import warnings

# Sklearn libraries
from sklearn.model_selection import train_test_split
```

```
from sklearn.feature_extraction.text import
    TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    classification_report,
    confusion_matrix,
    accuracy_score,
    roc_curve,
    roc_auc_score,
    precision_recall_curve,
    average_precision_score
)

# Configuration
warnings.filterwarnings('ignore')
sns.set_style("whitegrid")
plt.rcParams.update({'font.size': 12})

# ======================================================
# 1. DATA LOADING & PREPARATION
# ======================================

def load_real_data():
    """Downloads the Twitter US Airline Sentiment dataset
    ."""
    url = "https://raw.githubusercontent.com/
    satyajeetkrjha/kaggle-Twitter-US-Airline-Sentiment-/
    master/Tweets.csv"
    print(f"[INFO] Downloading dataset from: {url}")
    try:
        s = requests.get(url, timeout=10).content
        df = pd.read_csv(io.StringIO(s.decode('utf-8')))
        print(f"[INFO] Download success. Total rows: {len
    (df)}")
        return df
    except Exception as e:
        print(f"[ERROR] Download failed: {e}. Using
    synthetic data instead.")
        return None

def get_challenge_set():
    """Creates the specific 20-sentence test set
    mentioned in the requirement."""
    data = {
        'text': [
```

```
            # 10 Negative
            "The flight was delayed for 5 hours, terrible
     service.",
            "I lost my luggage and the staff was rude.",
            "Worst airline ever, never flying with you
    again.",
            "Food was cold and disgusting.",
            "Seats are uncomfortable and no leg room.",
            "Customer service put me on hold for an hour.
    ",
            "My flight was cancelled without notice.",
            "Very unhappy with the hidden fees.",
            "The plane was dirty and smelled bad.",
            "They broke my suitcase, very disappointed.",
            # 10 Positive
            "Great flight, landed on time!",
            "The crew was amazing and very helpful.",
            "Love this airline, best experience ever.",
            "Smooth landing and comfortable seats.",
            "Thank you for the upgrade, excellent service
    .",
            "Good food and great entertainment system.",
            "Quick check-in process, very efficient.",
            "The flight attendants were so nice and
    polite.",
            "Had a wonderful trip, thanks!",
            "Best value for money, will fly again."
        ],
        'ground_truth': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                         1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
     }
     return pd.DataFrame(data)

# --- EXECUTION: LOAD DATA ---
df = load_real_data()

# Preprocessing
print("\n" + "="*50)
print("STEP 1: DATA PREPROCESSING & EXPLORATION")
print("="*50)

# Filter only Positive and Negative (Binary
    Classification)
df_filtered = df[df['airline_sentiment'].isin(['positive'
    , 'negative'])].copy()
# Map to 0 and 1
```

```
df_filtered['label'] = df_filtered['airline_sentiment'].
    map({'negative': 0, 'positive': 1})

# Clean Text Function
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r"https?://\S+|www\.\S+", "", text) #
    Remove URLs
    text = re.sub(r"@\w+", "", text)                  #
    Remove Mentions
    text = re.sub(r"[^a-z\s]", "", text)              #
    Remove special chars
    return text.strip()

df_filtered['clean_text'] = df_filtered['text'].apply(
    clean_text)

# Dataset Statistics
n_pos = sum(df_filtered['label'] == 1)
n_neg = sum(df_filtered['label'] == 0)
print(f"Total Samples used: {len(df_filtered)}")
print(f" - Negative (0): {n_neg}")
print(f" - Positive (1): {n_pos}")
print(f" - Imbalance Ratio: 1 : {n_neg/n_pos:.2f}")

# Visualizing Class Distribution
plt.figure(figsize=(6, 4))
sns.countplot(x='airline_sentiment', data=df_filtered,
    palette='viridis')
plt.title("Class Distribution (Original Dataset)")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.show()

# ====================================
# 2. MODEL TRAINING (TF-IDF + LOGISTIC REGRESSION)
# ====================================
print("\n" + "="*50)
print("STEP 2: MODEL TRAINING")
print("="*50)

# Split Data
X_train, X_test, y_train, y_test = train_test_split(
    df_filtered['clean_text'],
    df_filtered['label'],
    test_size=0.2,
```

```
    random_state=42,
    stratify=df_filtered['label']
)

# Pipeline
model_pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1, 2),
   max_features=5000, stop_words='english')),
    ('clf', LogisticRegression(C=1.0, solver='liblinear',
    class_weight='balanced', random_state=42))
])

print("Training Logistic Regression with TF-IDF features
   ...")
model_pipeline.fit(X_train, y_train)
print("Training Completed.")

# ====================================
# 3. SCIENTIFIC EVALUATION (METRICS & PLOTS)
# ====================================
print("\n" + "="*50)
print("STEP 3: EVALUATION & VISUALIZATION")
print("="*50)

# Predictions
y_pred = model_pipeline.predict(X_test)
y_prob = model_pipeline.predict_proba(X_test)[:, 1]

# 3.1 Classification Report
print("--- Classification Report ---")
print(classification_report(y_test, y_pred, target_names
   =['Negative', 'Positive'], digits=4))

# 3.2 Metrics Calculation
acc = accuracy_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)
pr_auc = average_precision_score(y_test, y_prob)
print(f"Accuracy: {acc:.4f}")
print(f"ROC AUC : {roc_auc:.4f}")
print(f"PR AUC  : {pr_auc:.4f}")

# 3.3 PLOTTING
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# Plot A: Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
```

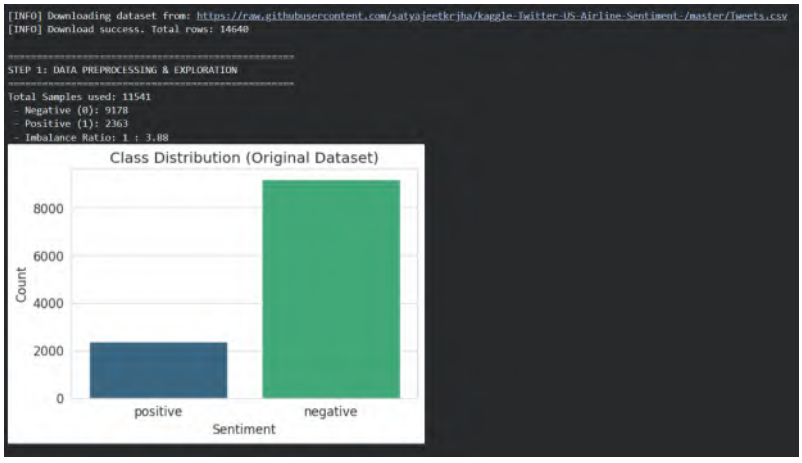
```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=
    axes[0], cbar=False,
            xticklabels=['Pred Neg', 'Pred Pos'],
    yticklabels=['True Neg', 'True Pos'])
axes[0].set_title("Confusion Matrix")

# Plot B: ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_prob)
axes[1].plot(fpr, tpr, color='darkorange', lw=2, label=f'
    AUC = {roc_auc:.3f}')
axes[1].plot([0, 1], [0, 1], color='navy', lw=2,
    linestyle='--')
axes[1].set_xlim([0.0, 1.0])
axes[1].set_ylim([0.0, 1.05])
axes[1].set_xlabel('False Positive Rate')
axes[1].set_ylabel('True Positive Rate')
axes[1].set_title('ROC Curve')
axes[1].legend(loc="lower right")

# Plot C: Precision-Recall Curve
prec, rec, _ = precision_recall_curve(y_test, y_prob)
axes[2].plot(rec, prec, color='green', lw=2, label=f'AP =
    {pr_auc:.3f}')
axes[2].set_xlabel('Recall')
axes[2].set_ylabel('Precision')
axes[2].set_title('Precision-Recall Curve')
axes[2].legend(loc="lower left")

plt.tight_layout()
plt.show()

# ====================================
# 4. FEATURE INTERPRETATION (TOP WORDS)
# ====================================
print("\n" + "="*50)
print("STEP 4: MODEL INTERPRETATION (TOP FEATURES)")
print("="*50)

feature_names = model_pipeline.named_steps['tfidf'].
    get_feature_names_out()
coefs = model_pipeline.named_steps['clf'].coef_[0]

# Get indices of top sorted coefficients
sorted_indices = np.argsort(coefs)
top_k = 10
```

```
top_neg_indices = sorted_indices[:top_k]
top_pos_indices = sorted_indices[-top_k:][::-1]

print(f"{'NEGATIVE WORDS':<20} | {'WEIGHT':<10} || {'
    POSITIVE WORDS':<20} | {'WEIGHT':<10}")
print("-" * 65)
for i in range(top_k):
    neg_word = feature_names[top_neg_indices[i]]
    neg_w = coefs[top_neg_indices[i]]

    pos_word = feature_names[top_pos_indices[i]]
    pos_w = coefs[top_pos_indices[i]]

    print(f"{neg_word:<20} | {neg_w:.4f}   || {pos_word
    :<20} | {pos_w:.4f}")

# ======================================
# 5. CHALLENGE: TESTING ON 20 MOCK SENTENCES
# ======================================
print("\n" + "="*50)
print("STEP 5: TESTING ON 20 CHALLENGE SENTENCES")
print("="*50)

# Get the challenge data
df_challenge = get_challenge_set()
df_challenge['clean_text'] = df_challenge['text'].apply(
    clean_text)

# Predict
df_challenge['pred_label'] = model_pipeline.predict(
    df_challenge['clean_text'])
df_challenge['pred_prob'] = model_pipeline.predict_proba(
    df_challenge['clean_text'])[:, 1]
df_challenge['sentiment_pred'] = df_challenge['pred_label
    '].map({0: 'NEGATIVE', 1: 'POSITIVE'})

# Display Results
print(f"Testing on {len(df_challenge)} new sentences...\n
    ")

# Color formatting for terminal (optional visual aid)
for i, row in df_challenge.iterrows():
    text = row['text']
    truth = "POSITIVE" if row['ground_truth'] == 1 else "
    NEGATIVE"
    pred = row['sentiment_pred']
```

```
    prob = row['pred_prob']

    # Check if correct
    is_correct = (row['ground_truth'] == row['pred_label'
   ])
    result_str = "[CORRECT]" if is_correct else "[WRONG]"

    print(f"{i+1:02d}. {text[:60]}...")
    print(f"    -> Truth: {truth} | Pred: {pred} ({prob
   :.4f}) | {result_str}")

# Final Score on Challenge Set
acc_challenge = accuracy_score(df_challenge['ground_truth
   '], df_challenge['pred_label'])
print(f"\nAccuracy on Challenge Set: {acc_challenge
   *100:.2f}%")
```

(e) Minh hoạ kết quả thực nghiệm và thảo luận

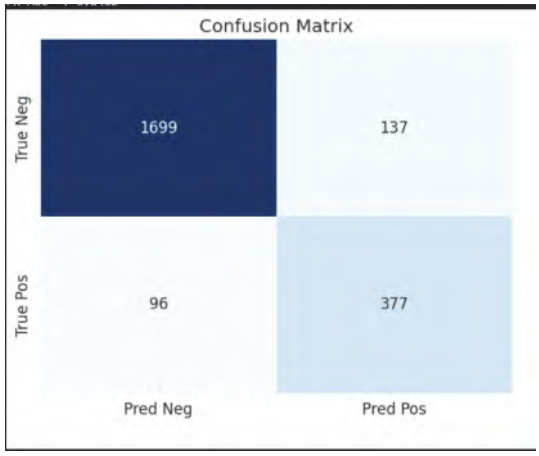
Các kết quả từ chương trình:



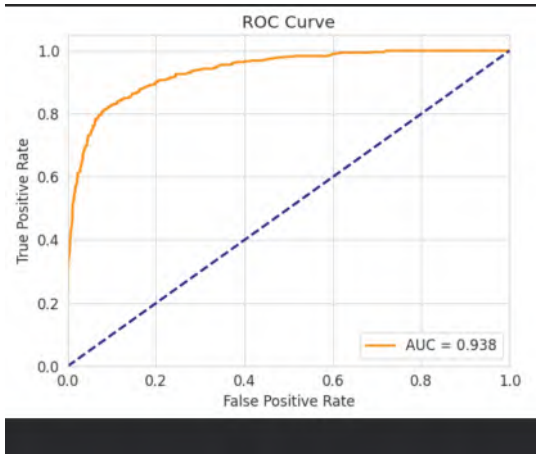
Hình 5: Kết quả bước 1

```
================================================
STEP 2: MODEL TRAINING
================================================
Training Logistic Regression with TF-IDF features...
Training Completed.

================================================
STEP 3: EVALUATION & VISUALIZATION
================================================
--- Classification Report ---
              precision    recall  f1-score   support

    Negative     0.9465    0.9254    0.9358      1836
    Positive     0.7335    0.7970    0.7639       473

    accuracy                         0.8991      2309
   macro avg     0.8400    0.8612    0.8499      2309
weighted avg     0.9029    0.8991    0.9006      2309

Accuracy: 0.8991
ROC AUC : 0.9376
PR AUC  : 0.8462
```

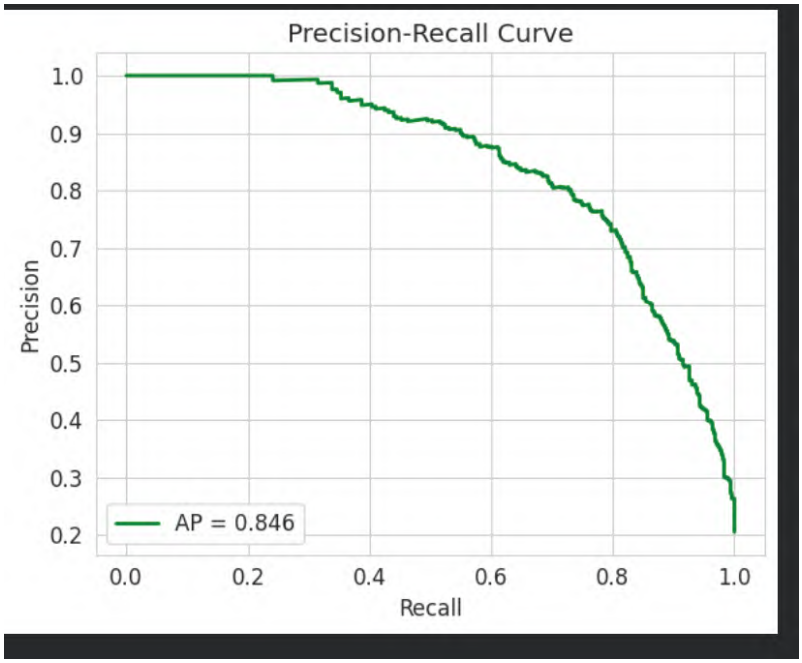
Hình 6: Kết quả Bước 2



Hình 7: Kết quả Bước 3



Hình 8: Đường cong ROC minh họa



Hình 9: Đường cong Precision-Recall minh họa

```
==================================================
STEP 4: MODEL INTERPRETATION (TOP FEATURES)
==================================================
NEGATIVE WORDS       | WEIGHT     || POSITIVE WORDS       | WEIGHT
-------------------------------------------------------------------
hours                | -3.9344    || thanks               | 9.0335
hold                 | -3.7097    || thank                | 8.6004
worst                | -3.6636    || great                | 5.9339
delayed              | -3.3565    || awesome              | 4.6739
hour                 | -3.1578    || love                 | 4.5184
bag                  | -2.6623    || amazing              | 4.3424
luggage              | -2.6338    || best                 | 4.1714
cancelled            | -2.6316    || thx                  | 3.4101
dont                 | -2.6308    || good                 | 3.2375
rude                 | -2.5004    || kudos                | 3.2003
```

Hình 10: Kết quả bước 4

```
================================================
STEP 5: TESTING ON 20 CHALLENGE SENTENCES
================================================
Testing on 20 new sentences...

01. The flight was delayed for 5 hours, terrible service....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.0106) | [CORRECT]
02. I lost my luggage and the staff was rude....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.0544) | [CORRECT]
03. Worst airline ever, never flying with you again....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.1277) | [CORRECT]
04. Food was cold and disgusting....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.3658) | [CORRECT]
05. Seats are uncomfortable and no leg room....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.1881) | [CORRECT]
06. Customer service put me on hold for an hour....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.0366) | [CORRECT]
07. My flight was cancelled without notice....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.0693) | [CORRECT]
08. Very unhappy with the hidden fees....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.3013) | [CORRECT]
09. The plane was dirty and smelled bad....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.1143) | [CORRECT]
10. They broke my suitcase, very disappointed....
    -> Truth: NEGATIVE | Pred: NEGATIVE (0.1186) | [CORRECT]
11. Great flight, landed on time!...
    -> Truth: POSITIVE | Pred: POSITIVE (0.8174) | [CORRECT]
12. The crew was amazing and very helpful....
    -> Truth: POSITIVE | Pred: POSITIVE (0.9725) | [CORRECT]
13. Love this airline, best experience ever....
    -> Truth: POSITIVE | Pred: POSITIVE (0.9607) | [CORRECT]
14. Smooth landing and comfortable seats....
    -> Truth: POSITIVE | Pred: POSITIVE (0.7994) | [CORRECT]
15. Thank you for the upgrade, excellent service....
    -> Truth: POSITIVE | Pred: POSITIVE (0.9856) | [CORRECT]
16. Good food and great entertainment system....
    -> Truth: POSITIVE | Pred: POSITIVE (0.9310) | [CORRECT]
17. Quick check-in process, very efficient....
    -> Truth: POSITIVE | Pred: POSITIVE (0.5278) | [CORRECT]
18. The flight attendants were so nice and polite....
    -> Truth: POSITIVE | Pred: POSITIVE (0.7128) | [CORRECT]
19. Had a wonderful trip, thanks!...
    -> Truth: POSITIVE | Pred: POSITIVE (0.9648) | [CORRECT]
20. Best value for money, will fly again....
    -> Truth: POSITIVE | Pred: POSITIVE (0.7182) | [CORRECT]

Accuracy on Challenge Set: 100.00%
```

Hình 11: Kết quả thử nghiệm trên 20 mẫu câu

Nhận xét về mô hình và kết quả thực nghiệm.
Trên bộ dữ liệu Twitter US Airline Sentiment sau khi lọc bỏ nhãn `neutral`, mô hình TF–IDF kết hợp Logistic Regression đạt độ chính xác toàn cục Accuracy ≈ 0.8991 trong bối cảnh phân bố lớp mất cân bằng khá mạnh (tỉ lệ `positive` : `negative` xấp xỉ $1:3.9$). Báo cáo phân loại cho thấy lớp `negative` đạt Precision/Recall/F1 tương ứng là $0.9465/0.9254/0.9358$ với 1836 mẫu, trong khi lớp `positive` đạt $0.7335/0.7970/0.7639$ với 473 mẫu. Ma trận nhầm lẫn (TN $= 1699$, FP $= 137$, FN $= 96$, TP $= 377$) khẳng định mô hình kiểm soát rất tốt các dự đoán sai dương (false positive) nhưng vẫn còn bỏ sót một phần không nhỏ các trường hợp tích cực (false negative), phản ánh chiến lược phân loại tương đối *bảo thủ* đối với lớp thiểu số.

Về năng lực phân tách, đường ROC nằm rõ ràng phía trên đường chéo ngẫu nhiên với ROC-AUC ≈ 0.938, cho thấy Logistic Regression khai thác khá hiệu quả thông tin

tuyến tính trong không gian TF–IDF trên toàn bộ miền ngưỡng. Đường Precision–Recall cho lớp tích cực có diện tích dưới đường cong PR-AUC ≈ 0.846, giá trị cao so với tỉ lệ xuất hiện lớp tích cực trong dữ liệu, biểu thị rằng mô hình duy trì được Precision tương đối tốt ngay cả khi tăng Recall. Danh sách các đặc trưng có trọng số lớn nhất cũng phù hợp với trực giác ngôn ngữ: các từ như `thanks`, `thank`, `great`, `awesome`, `love` mang trọng số dương lớn, trong khi `hours`, `worst`, `delayed`, `cancelled`, `rude` mang trọng số âm lớn, qua đó tăng mức độ minh bạch và khả năng giải thích của mô hình.

Cuối cùng, khi kiểm thử trên 20 câu bình luận giả định được xây dựng thủ công (10 câu tiêu cực và 10 câu tích cực với mức độ rõ ràng cao về ngữ nghĩa), mô hình đạt độ chính xác 100% và các xác suất dự đoán luôn lệch hẳn về phía nhãn đúng (xấp xỉ 0.01–0.36 cho những câu tiêu cực và 0.53–0.99 cho những câu tích cực). Đây là một chỉ báo tốt cho thấy mô hình đã học được các tín hiệu cảm xúc cơ bản; tuy nhiên, kết quả trên tập thử thách nhỏ và dễ này chỉ nên được xem như kiểm tra trực giác bổ sung, không thay thế cho các đánh giá định lượng chính thức trên tập kiểm tra chuẩn. Tổng thể, Logistic Regression trên đặc trưng TF–IDF cho thấy là một mô hình nền tảng mạnh, ổn định và dễ diễn giải cho bài toán phân tích cảm xúc các bình luận mạng xã hội.

Kết luận

Nội dung Chương này đã hướng dẫn chi tiết cho người học quy trình xây dựng một hệ thống phân tích cảm xúc trên dữ liệu mạng xã hội theo hướng tiếp cận nền tảng: biểu diễn văn bản bằng TF–IDF và huấn luyện mô hình Logistic Regression. Xuyên suốt nội dung chương, bài toán được diễn giải nhất quán bằng ngôn ngữ đại số tuyến tính: mỗi văn bản được ánh xạ thành véc-tơ đặc trưng $\boldsymbol{x}_i \in \mathbb{R}^d$, toàn bộ tập dữ liệu tương ứng với ma trận thưa $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, quá trình học mô hình quy về tối ưu hoá một hàm mục tiêu lồi dựa trên cross-entropy kèm chuẩn hoá ℓ_2. Từ đó, các phép toán cốt lõi của hệ thống: tính điểm số tuyến tính, biến đổi sigmoid, và cập nhật tham số qua gradient đều có thể mô tả gọn bằng các phép nhân ma trận–véc-tơ và các toán tử cơ bản trong giải tích.

Về mặt thực nghiệm, hệ thống được triển khai theo một pipeline chuẩn gồm tiền xử lý dữ liệu mạng xã hội, trích đặc trưng TF–IDF, huấn luyện Logistic Regression, đánh giá bằng các độ đo tiêu chuẩn và trực quan hoá kết quả. Kết quả cho thấy mô hình đạt hiệu năng cao trên tập kiểm tra với Accuracy $= 0.910$, ROC–AUC $= 0.9448$ và PR–AUC (Average Precision) $= 0.8630$, đồng thời có khả năng diễn giải thông qua các đặc trưng từ/cụm từ có trọng số lớn nhất. Tuy nhiên, sự mất cân bằng lớp làm nổi bật đánh đổi giữa Precision và Recall cho lớp tích cực, hàm ý rằng trong bối cảnh vận hành thực tế cần lựa chọn ngưỡng τ theo mục tiêu ứng dụng và đánh giá bằng các độ đo nhạy với lớp thiểu số.

Trên nền tảng đã xây dựng, các hướng phát triển hợp lý gồm: (i) cải thiện biểu diễn

bằng đặc trưng ký tự, mở rộng n-gram hoặc chuyển sang các biểu diễn ngữ nghĩa (word embeddings, Transformer) để nắm bắt ngữ cảnh sâu; (ii) hoàn thiện quy trình triển khai với lưu pipeline, kiểm soát rò rỉ dữ liệu, theo dõi trôi dữ liệu và tái huấn luyện định kỳ nhằm đảm bảo mô hình ổn định khi phân phối dữ liệu mạng xã hội thay đổi theo thời gian.

Chương 5

BÀI TẬP NÂNG CAO VỀ ĐẠI SỐ TUYẾN TÍNH

ỨNG DỤNG TRONG AI VÀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Giới thiệu 172
1 Hình học không gian nhúng (embedding) và phép toán vector 172
1.1. Bài tập 1. Số học vector trong không gian nhúng (embedding) và bài toán tương tự (analogy) 172
1.2. Bài tập 2. Nhúng câu (sentence embedding): trung bình, dịch chuyển (translation) và vai trò độ lệch (bias) 173
1.3. Bài tập 3. Chiếu lên không gian con chủ đề bằng bình phương tối thiểu 173
1.4. Bài tập 4. Trực chuẩn hóa hướng đặc trưng bằng Gram–Schmidt 174
1.5. Bài tập 5. Ma trận quay và nhúng vị trí quay (Rotary Positional Embedding, RoPE) 174
1.6. Bài tập 6. Tích Hadamard và cơ chế cổng (gating) trong GRU 175
2 Biến đổi tuyến tính và tính ổn định số trong học máy 175
2.1. Bài tập 7. Biến đổi afin (affine) như nhân ma trận mở rộng . 175
2.2. Bài tập 8. Kết nối dư (residual connection), ma trận đơn vị và phổ trị riêng 176
2.3. Bài tập 9. Thăm dò tuyến tính (linear probing) dưới dạng bài toán bình phương tối thiểu (least squares, chuẩn Frobenius) . 176
2.4. Bài tập 10. Hồi quy tuyến tính Ridge (Ridge regression) và nghịch đảo ổn định 177
2.5. Bài tập 11. Tự chú ý (self-attention) dưới dạng ma trận: chuyển vị (transpose), mặt nạ (mask) và độ phức tạp 177
2.6. Bài tập 12. Số điều kiện và chiến lược xử lý kém điều kiện (ill-conditioning) 178
3 Hiệp phương sai, phân rã riêng và làm trắng đặc trưng 178
3.1. Bài tập 13. Ma trận hiệp phương sai của embedding câu và ý nghĩa 178
3.2. Bài tập 14. Whitening embedding bằng phân rã trị riêng . . 179
3.3. Bài tập 15. Spectral clustering trên đồ thị đồng xuất hiện từ . 179
3.4. Bài tập 16. Giảm hiện tượng các vector bị dồn về một vài hướng trội (anisotropy) bằng cách loại bỏ các hướng chính . 179
3.5. Bài tập 17. Phương pháp lặp luỹ thừa (power iteration) để xấp xỉ trị riêng lớn nhất. 180
3.6. Bài tập 18. Ổn định của RNN tuyến tính hóa qua trị riêng . . 180
4 SVD và xấp xỉ hạng thấp để nén và căn chỉnh mô hình 181

4.1. Bài tập 19. Xấp xỉ hạng thấp cho ma trận embedding từ . . . 181
4.2. Bài tập 20. Căn chỉnh embedding song ngữ bằng bài toán Procrustes trực giao (Orthogonal Procrustes) 181
4.3. Bài tập 21. Phân rã hạng thấp cho lớp chiếu ra từ vựng . . . 182
4.4. Bài tập 22. Giả nghịch đảo Moore–Penrose và nghiệm chuẩn nhỏ nhất 182
4.5. Bài tập 23. Chuẩn Frobenius, chuẩn hạt nhân và liên hệ với hạng 183
4.6. Bài tập 24. LoRA như cập nhật hạng thấp và liên hệ với SVD 183
5 Tensor và tích chập nâng cao trong kiến trúc hiện đại 184
5.1. Bài tập 25. Tích ngoài và tương tác bậc hai trong mô hình hỏi–đáp 184
5.2. Bài tập 26. Co tensor (tensor contraction) cho Multi-Head Attention và exttteinsum 184
5.3. Bài tập 27. phép nhân tensor theo mode n (mode-n product) cho lớp tuyến tính áp dụng trên mọi token 185
5.4. Bài tập 28. Trải phẳng/tái định dạng (flatten/reshape) giữa CNN và MLP: kiểm soát kích thước 185
5.5. Bài tập 29. Tích chập nhân quả (causal) và giãn cách (dilation): trường tiếp nhận (receptive field) trên chuỗi . . . 186
5.6. Bài tập 30. Tích chập tách theo chiều sâu (depthwise separable convolution) và đếm tham số 186
Kết luận 187

Giới thiệu

Chương này gồm 30 bài tập mở rộng dựa trên các khái niệm nền tảng của Chương 1 (vector, ma trận, trị riêng–vector riêng, SVD, tensor và tích chập) và kiến thức thực hành được trình bày ở các Chương 2-4. Hệ thống các bài tập trong chương này được mở rộng ở nhiều khía cạnh ứng dụng Đại số tuyến tính cho các bài toán trong AI và NLP hiện đại như: căn chỉnh không gian embedding, làm trắng (whitening) đặc trưng, ổn định của mạng hồi quy, xấp xỉ hạng thấp để nén mô hình, và các phép toán tensor để mô tả cơ chế attention.

Một điểm khác biệt lớn ở nội dung chương này là mỗi bài chỉ cung cấp yêu cầu bài toán và hướng dẫn, gợi ý nhưng không có lời giải chi tiết để người học tự triển khai trên nền tảng tiếp thu lý thuyết và thực hành ứng dụng Đại số tuyến tính đã học ở các Chương trước và tự kiểm tra tính đúng đắn bằng các phép thử số học và trực quan hóa.

1 Hình học không gian nhúng (embedding) và phép toán vector

1.1. Bài tập 1. Số học vector trong không gian nhúng (embedding) và bài toán tương tự (analogy)

Đề bài.

Cho các vector embedding (đã được huấn luyện) trong $\mathbb{R}^d$:

$\vec{e}(\text{“king”})$, $\vec{e}(\text{“man”})$, $\vec{e}(\text{“woman”})$, $\vec{e}(\text{“queen”})$.

(a) Xây dựng vector truy vấn $\vec{q} = \vec{e}(\text{“king”}) - \vec{e}(\text{“man”}) + \vec{e}(\text{“woman”})$.

(b) Dùng độ tương đồng cosine để xếp hạng k từ gần $\vec{q}$ nhất trong từ vựng.

(c) Phân tích vì sao phép tịnh tiến trong không gian embedding có thể mã hóa quan hệ ngữ nghĩa.

Hướng dẫn và gợi ý.

- Nhắc lại: phép cộng/trừ vector tương ứng với tịnh tiến trong không gian; hãy diễn giải $\vec{q}$ như “king” đã loại bỏ thuộc tính giới và thêm thuộc tính giới nữ.
- Để xếp hạng, chuẩn hóa các vector rồi tính $\cos(\vec{q}, \vec{e}(w)) = \frac{\vec{q}^{\top}\vec{e}(w)}{\|\vec{q}\|\|\vec{e}(w)\|}$ cho mọi từ w.
- Trong thực nghiệm, chúng tôi loại bỏ khỏi danh sách ứng viên các từ đã được dùng để tạo $\vec{q}$, nhằm tránh hiện tượng *láng giềng gần tầm thường* (trivial nearest neighbor) (tức là trường hợp khi tìm phần tử gần nhất cho $\vec{q}$ thì kết

quả lại rơi vào chính các từ đã dùng để tạo ra $\vec{q}$, nên hiển nhiên, không mang thông tin mới và làm méo đánh giá). .

- Thảo luận các điều kiện khiến phép tương tự đúng/không đúng (chất lượng embedding, nhiễu, đa nghĩa).

1.2. Bài tập 2. Nhúng câu (sentence embedding): trung bình, dịch chuyển (translation) và vai trò độ lệch (bias)

Đề bài. Giả sử câu S gồm L token với embedding $\vec{x}_1, \ldots, \vec{x}_L \in \mathbb{R}^d$. Xét hai cách tạo vector câu:

$$\vec{s}_{\text{mean}} = \frac{1}{L}\sum_{i=1}^{L} \vec{x}_i, \qquad \vec{s}_{\text{cls}} = \vec{x}_{[\text{CLS}]}.$$

(a) Chứng minh $\vec{s}_{\text{mean}}$ bất biến theo hoán vị thứ tự token (bag-of-embeddings).
(b) Cho một vector độ lệch (bias) $\vec{b} \in \mathbb{R}^d$ và xét $\tilde{\vec{x}}_i = \vec{x}_i + \vec{b}$. Tính $\tilde{\vec{s}}_{\text{mean}}$ theo $\vec{s}_{\text{mean}}$ và $\vec{b}$.
(c) Bàn luận ý nghĩa của phép tịnh tiến bởi $\vec{b}$ trong không gian đặc trưng (liên hệ đến hạng tử độ lệch (bias) của lớp tuyến tính).

Hướng dẫn và gợi ý.

- Dùng tính giao hoán của phép cộng vector để xử lý (a).
- Ở (b), khai triển tổng: $\frac{1}{L}\sum(\vec{x}_i + \vec{b}) = \vec{s}_{\text{mean}} + \vec{b}$.
- Liên hệ (c) với biến đổi affine $\vec{y} = W\vec{x} + \vec{b}$ và vai trò dịch chuyển tâm dữ liệu.

1.3. Bài tập 3. Chiếu lên không gian con chủ đề bằng bình phương tối thiểu

Đề bài. Trong hệ thống phân loại chủ đề, mỗi chủ đề được gán một vector "prototype" $\vec{t}_1, \vec{t}_2, \vec{t}_3 \in \mathbb{R}^d$ (không nhất thiết trực giao). Với embedding câu $\vec{s} \in \mathbb{R}^d$:
(a) Lập ma trận $T = [\vec{t}_1\ \vec{t}_2\ \vec{t}_3] \in \mathbb{R}^{d\times 3}$ và xét bài toán xấp xỉ theo nghĩa bình phương tối thiểu:

$$\min_{\vec{\alpha}\in\mathbb{R}^3} \|\vec{s} - T\vec{\alpha}\|_2^2.$$

Viết hệ phương trình chuẩn (normal equation) cho $\vec{\alpha}$.
(b) Nếu $\vec{t}_1, \vec{t}_2, \vec{t}_3$ trực giao và đã chuẩn hóa, hãy nêu công thức đơn giản của α_j.
(c) Nêu cách dùng $\vec{\alpha}$ để quyết định chủ đề chính của câu.

Hướng dẫn và gợi ý.

- Nhắc lại: bài toán least squares dẫn đến $(T^\top T)\vec{\alpha} = T^\top \vec{s}$ khi $T^\top T$ khả nghịch.
- Trường hợp trực giao chuẩn hóa: $T^\top T = I$ nên $\vec{\alpha} = T^\top \vec{s}$; khi đó $\alpha_j = \vec{t}_j^{\top}\vec{s}$.

- Diễn giải α_j như "mức chiếu" của $\vec{s}$ lên hướng chủ đề $\vec{t}_j$; chọn j có $|\alpha_j|$ lớn nhất (hoặc lớn nhất nếu $\alpha_j \geq 0$).

1.4. Bài tập 4. Trực chuẩn hóa hướng đặc trưng bằng Gram–Schmidt

Đề bài. Cho ba vector đặc trưng $\vec{u}_1, \vec{u}_2, \vec{u}_3 \in \mathbb{R}^d$ biểu diễn ba thuộc tính (ví dụ: cảm xúc, chủ đề, độ trang trọng) trích từ mô hình NLP.
(a) Mô tả thuật toán Gram–Schmidt để tạo hệ trực chuẩn $\{\vec{q}_1, \vec{q}_2, \vec{q}_3\}$.
(b) Giải thích vì sao việc trực chuẩn hóa các hướng đặc trưng có ích khi phân tích/giải thích mô hình (interpretability).
(c) Với một embedding câu $\vec{s}$, hãy nêu cách tách $\vec{s}$ thành tổng các thành phần chiếu lên $\vec{q}_1, \vec{q}_2, \vec{q}_3$ và phần dư.

Hướng dẫn và gợi ý.

- Nhắc lại: $\vec{q}_1 = \frac{\vec{u}_1}{\|\vec{u}_1\|}$; sau đó loại bỏ thành phần theo $\vec{q}_1$ khỏi $\vec{u}_2$, v.v.
- Trong (b), nhấn mạnh tính không chồng chéo (decorrelation) giữa các hướng giúp diễn giải từng thuộc tính độc lập hơn.
- Trong (c), dùng phép chiếu: $\text{proj}_{\vec{q}_j}(\vec{s}) = (\vec{s}^\top \vec{q}_j)\vec{q}_j$; phần dư $\vec{r} = \vec{s} - \sum_j \text{proj}_{\vec{q}_j}(\vec{s})$.

1.5. Bài tập 5. Ma trận quay và nhúng vị trí quay (Rotary Positional Embedding, RoPE)

Đề bài. Trong Rotary Positional Embedding (RoPE), mỗi cặp chiều $(2k, 2k+1)$ của vector được quay một góc phụ thuộc vị trí. Xét một cặp toạ độ (x_{2k}, x_{2k+1}) và góc θ_p tại vị trí p.
(a) Viết ma trận quay $R(\theta_p) \in \mathbb{R}^{2\times 2}$ và biểu diễn $(\tilde{x}_{2k}, \tilde{x}_{2k+1})^\top = R(\theta_p)(x_{2k}, x_{2k+1})^\top$.
(b) Chứng minh $R(\theta_p)$ trực giao và bảo toàn chuẩn Euclid của cặp toạ độ.
(c) Giải thích trực giác: vì sao *quay* trong không gian con 2 chiều có thể mã hóa thông tin vị trí mà vẫn giữ được cấu trúc hình học của embedding.

Hướng dẫn và gợi ý.

- Ma trận quay chuẩn: $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$.
- Kiểm tra trực giao bằng cách tính $R(\theta)^\top R(\theta) = I$; từ đó suy ra bảo toàn chuẩn.
- Liên hệ RoPE với việc biến đổi tương đối: tích vô hướng giữa các vector đã quay phụ thuộc vào hiệu vị trí (gợi ý: dùng công thức lượng giác).

1.6. Bài tập 6. Tích Hadamard và cơ chế cổng (gating) trong GRU

Đề bài. Trong GRU, cập nhật trạng thái thường có dạng

$$\vec{h}_t = (1-\vec{z}_t)\odot\vec{h}_{t-1} + \vec{z}_t\odot\tilde{\vec{h}}_t,$$

với $\odot$ là phép nhân Hadamard (nhân theo từng phần tử), và $\vec{z}_t \in (0,1)^d$ là cổng cập nhật (update gate).
(a) Với $d=4$, cho $\vec{h}_{t-1} = [0.2,\ -0.1,\ 0.7,\ 0.0]$ và $\tilde{\vec{h}}_t = [0.9,\ 0.3,\ 0.1,\ -0.2]$, hãy tính $\vec{h}_t$ khi $\vec{z}_t = [0.0,\ 0.5,\ 1.0,\ 0.2]$.
(b) Phân tích vai trò của tích Hadamard trong cơ chế gating.
(c) Đề xuất một kiểm thử số (sanity check) để xác nhận $0 \le z_{t,i} \le 1$ sẽ khiến $\vec{h}_t$ là tổ hợp lồi theo từng chiều.

Hướng dẫn và gợi ý.

- Thực hiện (a) theo từng chiều i: $h_{t,i} = (1-z_{t,i})h_{t-1,i} + z_{t,i}\tilde{h}_{t,i}$.
- Trong (b), nhấn mạnh: Hadamard cho phép điều khiển *mức mở* khác nhau cho từng chiều đặc trưng.
- Trong (c), gợi ý: chứng minh $h_{t,i}$ nằm giữa $h_{t-1,i}$ và $\tilde{h}_{t,i}$ khi $z_{t,i} \in [0,1]$.

2 Biến đổi tuyến tính và tính ổn định số trong học máy

2.1. Bài tập 7. Biến đổi afin (affine) như nhân ma trận mở rộng

Đề bài. Xét một lớp tuyến tính trong mạng NLP: $\vec{y} = W\vec{x} + \vec{b}$ với $W \in \mathbb{R}^{m\times d}$, $\vec{x} \in \mathbb{R}^d$, $\vec{b} \in \mathbb{R}^m$.
(a) Dùng thủ thuật mở rộng (augmentation) để viết phép biến đổi affine dưới dạng nhân ma trận thuần tuý $\vec{y} = \tilde{W}\tilde{\vec{x}}$ với $\tilde{W} \in \mathbb{R}^{m\times(d+1)}$.
(b) Cho một batch $X \in \mathbb{R}^{B\times d}$, hãy viết dạng vector hóa để tính $Y \in \mathbb{R}^{B\times m}$ trong một phép nhân ma trận.
(c) Trong bối cảnh fine-tuning mô hình ngôn ngữ, giải thích vì sao việc tách riêng vai trò của W và $\vec{b}$ giúp phân tích hiện tượng dịch chuyển phân phối (distribution shift).

Hướng dẫn và gợi ý.

- Thêm một chiều hằng số: $\tilde{\vec{x}} = [\vec{x}; 1]$, và $\tilde{W} = [W\ \vec{b}]$.
- Với batch, dùng $Y = XW^\top + \mathbf{1}\vec{b}^\top$ (trong đó $\mathbf{1} \in \mathbb{R}^B$ là vector toàn 1).
- Trong (c), gợi ý liên hệ: W thay đổi hướng/độ giãn của không gian, còn $\vec{b}$ tịnh tiến tâm biểu diễn.

2.2. Bài tập 8. Kết nối dư (residual connection), ma trận đơn vị và phổ trị riêng

Đề bài. Xét khối residual đơn giản: $\vec{y} = \vec{x} + F(\vec{x})$. Giả sử $F(\vec{x}) = A\vec{x}$ với $A \in \mathbb{R}^{d\times d}$ (xấp xỉ tuyến tính cục bộ).
(a) Viết $\vec{y}$ dưới dạng $\vec{y} = (I + A)\vec{x}$ và nêu ý nghĩa của ma trận đơn vị I.
(b) Nếu λ là trị riêng của A, hãy biểu diễn trị riêng tương ứng của $(I + A)$.
(c) Thảo luận điều kiện trên phổ (spectrum) của A để tránh bùng nổ/tiêu biến chuẩn của tín hiệu khi xếp chồng nhiều khối residual.

Hướng dẫn và gợi ý.

- Nhắc lại: I giữ nguyên $\vec{x}$ (skip connection).
- Dùng tính chất: trị riêng của $I + A$ là $1 + \lambda$ (với cùng vector riêng).
- Gợi ý (c): xem xét $\|(I + A)^L\|$ khi lặp L lần; trực giác: cần $|1 + \lambda|$ không quá lớn và không quá nhỏ.

2.3. Bài tập 9. Thăm dò tuyến tính (linear probing) dưới dạng bài toán bình phương tối thiểu (least squares, chuẩn Frobenius)

Đề bài. Trong "linear probing" cho mô hình ngôn ngữ, ta cố định encoder và học một bộ phân loại tuyến tính. Cho ma trận đặc trưng $X \in \mathbb{R}^{N\times d}$ (mỗi hàng là embedding của một câu) và nhãn one-vs-rest $Y \in \mathbb{R}^{N\times C}$. Xét bài toán bình phương tối thiểu:

$$\min_{W\in\mathbb{R}^{d\times C}} \|XW - Y\|_F^2.$$

(a) Viết điều kiện tối ưu bậc nhất dẫn đến hệ phương trình cho W.
(b) Giả sử X có hạng đầy đủ theo cột, hãy nêu nghiệm đóng của W.
(c) Định nghĩa ma trận chiếu $P = X(X^\top X)^{-1}X^\top$. Hãy giải thích ý nghĩa của PX và PY trong ngữ cảnh chiếu lên không gian cột (project onto the column space).

Hướng dẫn và gợi ý.

- Nhắc lại: với chuẩn Frobenius, có thể lấy đạo hàm theo W và đặt bằng 0 để được $X^\top XW = X^\top Y$.
- Nếu $X^\top X$ khả nghịch: $W = (X^\top X)^{-1}X^\top Y$.
- Ma trận P là phép chiếu trực giao lên không gian cột của X; từ đó $PX = X$ và PY là thành phần của Y có thể biểu diễn bởi các đặc trưng X.

2.4. Bài tập 10. Hồi quy tuyến tính Ridge (Ridge regression) và nghịch đảo ổn định

Đề bài. Tiếp tục Bài 9, xét Ridge regression (Tikhonov):

$$\min_W \ \|XW - Y\|_F^2 + \lambda \|W\|_F^2, \qquad \lambda > 0.$$

(a) Suy ra hệ phương trình cho nghiệm tối ưu W_λ.
(b) So sánh W_λ với nghiệm bình phương tối thiểu không điều chuẩn khi $X^\top X$ gần suy biến.
(c) Với một giá trị λ nhỏ và lớn, hãy dự đoán định tính ảnh hưởng đến độ khớp (fit) và khả năng tổng quát hóa (generalization) trong NLP.

Hướng dẫn và gợi ý.

- Điều kiện tối ưu: $(X^\top X + \lambda I)W = X^\top Y$.
- Gợi ý (b): λI cải thiện điều kiện số (conditioning), khiến ma trận luôn khả nghịch.
- Trong (c), liên hệ đánh đổi độ lệch–phương sai (bias–variance trade-off); λ lớn làm co trọng số về 0.

2.5. Bài tập 11. Tự chú ý (self-attention) dưới dạng ma trận: chuyển vị (transpose), mặt nạ (mask) và độ phức tạp

Đề bài. Xét self-attention một đầu (single-head) với $Q, K, V \in \mathbb{R}^{L \times d}$:

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}} + M\right), \qquad \mathrm{Attn}(Q,K,V) = AV,$$

trong đó $M \in \mathbb{R}^{L \times L}$ là ma trận mask (ví dụ: causal mask).
(a) Kiểm tra kích thước của từng biểu thức và giải thích vì sao $K^\top$ xuất hiện.
(b) Mô tả cách xây dựng M cho causal attention sao cho token ở vị trí i không nhìn thấy vị trí $j > i$.
(c) Ước lượng độ phức tạp tính toán theo L và d, và thảo luận vì sao việc giảm hạng (low-rank) hoặc sparse attention là cần thiết khi L lớn.

Hướng dẫn và gợi ý.

- Dùng quy tắc nhân ma trận: $QK^\top \in \mathbb{R}^{L \times L}$ là ma trận điểm giống nhau (similarity) giữa mọi cặp vị trí.
- Với causal mask, đặt $M_{ij} = -\infty$ (hoặc một số âm rất lớn) nếu $j > i$, và 0 nếu $j \le i$.
- Độ phức tạp: tính $QK^\top$ tốn $O(L^2 d)$, nhân AV tốn $O(L^2 d)$. Hãy liên hệ với điểm nghẽn (bottleneck) bộ nhớ $O(L^2)$.

2.6. Bài tập 12. Số điều kiện và chiến lược xử lý kém điều kiện (ill-conditioning)

Đề bài. Cho ma trận vuông khả nghịch $B \in \mathbb{R}^{d\times d}$ (ví dụ: ma trận Fisher/Gram trong huấn luyện).
(a) Định nghĩa số điều kiện (condition number) theo chuẩn 2: $\kappa_2(B) = \|B\|_2\|B^{-1}\|_2$.
(b) Giải thích ý nghĩa của $\kappa_2(B)$ đối với độ nhạy của nghiệm hệ $B\vec{x} = \vec{c}$ khi $\vec{c}$ bị nhiễu.
(c) Trong thực hành ML, nêu ít nhất hai chiến lược để giảm tình trạng kém điều kiện (ill-conditioning) khi cần giải hệ hoặc tính nghịch đảo xấp xỉ.

Hướng dẫn và gợi ý.

- Nhắc lại: với chuẩn 2, $\kappa_2(B) = \frac{\sigma_{\max}(B)}{\sigma_{\min}(B)}$ (liên hệ với SVD).
- Ý nghĩa (b): sai số tương đối của nghiệm có thể bị khuếch đại xấp xỉ bởi $\kappa_2(B)$.
- Chiến lược (c) gợi ý: chính quy Ridge (Ridge regularization) $(B + \lambda I)$, chuẩn hóa dữ liệu, dùng phân rã số ổn định (QR/SVD) thay vì nghịch đảo trực tiếp, hoặc tiền điều kiện hóa (preconditioning).

3 Hiệp phương sai, phân rã riêng và làm trắng đặc trưng

3.1. Bài tập 13. Ma trận hiệp phương sai của embedding câu và ý nghĩa

Đề bài. Bạn có N vector embedding câu $\vec{s}_1, \ldots, \vec{s}_N \in \mathbb{R}^d$. Gọi $\bar{\vec{s}} = \frac{1}{N}\sum_{i=1}^{N} \vec{s}_i$.
(a) Viết công thức ma trận hiệp phương sai $C \in \mathbb{R}^{d\times d}$ của tập embedding.
(b) Giải thích ý nghĩa của phần tử C_{pq} (liên hệ tương quan giữa hai chiều đặc trưng p, q).
(c) Đề xuất một cách trực quan hóa C (hoặc một phần của C) để phát hiện các chiều dư thừa trong embedding.

Hướng dẫn và gợi ý.

- Công thức chuẩn: $C = \frac{1}{N}\sum_{i=1}^{N}(\vec{s}_i - \bar{\vec{s}})(\vec{s}_i - \bar{\vec{s}})^\top$ (hoặc $\frac{1}{N-1}$).
- Nếu $|C_{pq}|$ lớn, hai chiều thay đổi cùng nhau; dấu của C_{pq} cho biết đồng biến hay nghịch biến.
- Trực quan hóa: heatmap cho ma trận con $k \times k$; hoặc xem phổ trị riêng của C để đo tập trung phương sai.

3.2. Bài tập 14. Whitening embedding bằng phân rã trị riêng

Đề bài. Xét ma trận hiệp phương sai C ở Bài 13 (giả sử đối xứng và xác định dương). Ta muốn làm trắng đặc trưng:

$$\vec{z} = W(\vec{s} - \bar{\vec{s}}), \qquad \text{sao cho} \quad \mathrm{Cov}(\vec{z}) \approx I.$$

(a) Nếu $C = U\Lambda U^\top$ là phân rã trị riêng, hãy nêu một lựa chọn W theo U, Λ.
(b) Giải thích vì sao cần xử lý trị riêng rất nhỏ (gợi ý: thêm ε).
(c) Thảo luận tác động của whitening lên độ tương đồng cosine giữa các câu (có thể tăng/giảm? vì sao?).

Hướng dẫn và gợi ý.

- Một lựa chọn phổ biến: $W = \Lambda^{-1/2}U^\top$ (hoặc $U\Lambda^{-1/2}U^\top$ tuỳ quy ước).
- Trị riêng nhỏ gây nổ số khi lấy nghịch đảo căn bậc hai; dùng $\Lambda_\varepsilon = (\Lambda + \varepsilon I)$.
- Whitening thay đổi hình học: các hướng phương sai lớn bị nén, hướng phương sai nhỏ được khuếch đại; vì vậy cosine có thể thay đổi đáng kể.

3.3. Bài tập 15. Spectral clustering trên đồ thị đồng xuất hiện từ

Đề bài. Xây dựng đồ thị đồng xuất hiện từ (word co-occurrence graph) với n từ trong từ vựng. Gọi $A \in \mathbb{R}^{n\times n}$ là ma trận kề (đối xứng, $A_{ij} \geq 0$), D là ma trận bậc với $D_{ii} = \sum_j A_{ij}$.
(a) Viết Laplacian chuẩn hóa $L_{\text{sym}} = I - D^{-1/2}AD^{-1/2}$.
(b) Nêu quy trình phân cụm phổ (spectral clustering) để chia từ vựng thành k cụm dựa trên k vector riêng ứng với k trị riêng nhỏ nhất của L_{sym}.
(c) Giải thích ý nghĩa ngữ nghĩa: vì sao các từ hay đi cùng nhau có xu hướng rơi vào cùng một cụm.

Hướng dẫn và gợi ý.

- Nhắc lại: với đồ thị liên thông, trị riêng nhỏ nhất của L_{sym} là 0; các vector riêng kế tiếp mang thông tin phân cụm.
- Quy trình (b) gợi ý: lấy ma trận $U_k \in \mathbb{R}^{n\times k}$ gồm k vector riêng, chuẩn hóa từng hàng, rồi chạy k-means trên các hàng.
- Diễn giải (c) theo tiêu chí độ trơn (smoothness): vector nhãn cụm biến thiên chậm trên các cạnh có trọng số lớn.

3.4. Bài tập 16. Giảm hiện tượng các vector bị dồn về một vài hướng trội (anisotropy) bằng cách loại bỏ các hướng chính

Đề bài. Hiện tượng *anisotropy* trong embedding câu: nhiều vector tập trung quanh một vài hướng chính, làm giảm khả năng phân biệt bằng cosine. Cho ma trận

embedding đã được trung tâm hóa $S \in \mathbb{R}^{N\times d}$.
(a) Gọi $\vec{u}_1$ là vector riêng ứng với trị riêng lớn nhất của $C = \frac{1}{N}S^\top S$. Viết phép khử thành phần theo $\vec{u}_1$: $\vec{s}' = \vec{s} - (\vec{s}^\top \vec{u}_1)\vec{u}_1$.
(b) Tổng quát hóa khi loại bỏ r hướng chính.
(c) Đề xuất một thước đo đơn giản để kiểm tra anisotropy trước/sau khi khử (gợi ý: phân bố cosine giữa các cặp câu ngẫu nhiên).

Hướng dẫn và gợi ý.

- Liên hệ: $S^\top S$ là Gram matrix theo chiều đặc trưng; $\vec{u}_1$ là hướng giải thích phương sai lớn nhất.
- Khi loại bỏ r hướng: dùng ma trận $U_r = [\vec{u}_1, \ldots, \vec{u}_r]$ và chiếu: $\vec{s}' = \vec{s} - U_r U_r^\top \vec{s}$.
- Thước đo (c): so sánh trung bình/độ lệch chuẩn của cosine giữa các cặp câu, anisotropy mạnh thường làm cosine bị lệch về giá trị cao.

3.5. Bài tập 17. Phương pháp lặp luỹ thừa (power iteration) để xấp xỉ trị riêng lớn nhất.

Đề bài. Cho ma trận đối xứng $C \in \mathbb{R}^{d\times d}$ (ví dụ: hiệp phương sai embedding). Ta muốn xấp xỉ trị riêng lớn nhất $\lambda_{\max}$ và vector riêng tương ứng bằng phương pháp lặp lũy thừa.
(a) Mô tả thuật toán lặp lũy thừa bắt đầu từ $\vec{v}_0 \neq \vec{0}$.
(b) Nêu điều kiện hội tụ (liên hệ khoảng cách giữa $\lambda_{\max}$ và trị riêng lớn thứ hai).
(c) Gợi ý cách ước lượng $\lambda_{\max}$ sau khi có $\vec{v}_t$ (dùng Rayleigh quotient (thương Rayleigh)).

Hướng dẫn và gợi ý.

- Thuật toán: lặp $\vec{v}_{t+1} = \frac{C\vec{v}_t}{\|C\vec{v}_t\|}$.
- Hội tụ nhanh khi $|\lambda_2/\lambda_1|$ nhỏ (với $|\lambda_1| = |\lambda_{\max}|$).
- Thương Rayleigh: $\hat{\lambda}_t = \frac{\vec{v}_t^\top C\vec{v}_t}{\vec{v}_t^\top \vec{v}_t}$.

3.6. Bài tập 18. Ổn định của RNN tuyến tính hóa qua trị riêng

Đề bài. Xét RNN tuyến tính hóa: $\vec{h}_t = W\vec{h}_{t-1}$ với $W \in \mathbb{R}^{d\times d}$.
(a) Viết $\vec{h}_t = W^t\vec{h}_0$ và liên hệ tốc độ tăng/giảm của $\|\vec{h}_t\|$ với phổ trị riêng của W.
(b) Nếu W chéo hóa được: $W = V\Lambda V^{-1}$, hãy biểu diễn W^t.
(c) Bàn luận vì sao chuẩn phổ (spectral radius) $\rho(W) = \max_i |\lambda_i|$ liên quan trực tiếp đến hiện tượng *vanishing/exploding gradients* trong RNN.

Hướng dẫn và gợi ý.

- Nhắc lại: nếu $\rho(W) < 1$ thì $W^t \to 0$ khi $t \to \infty$; nếu $\rho(W) > 1$ thì có thể bùng nổ.
- Nếu $W = V\Lambda V^{-1}$ thì $W^t = V\Lambda^t V^{-1}$ với Λ^t là lũy thừa từng phần tử trên đường chéo.
- Liên hệ gradient: đạo hàm lan truyền qua thời gian chứa tích các Jacobian, trong trường hợp tuyến tính chính là W^t.

4 SVD và xấp xỉ hạng thấp để nén và căn chỉnh mô hình

4.1. Bài tập 19. Xấp xỉ hạng thấp cho ma trận embedding từ

Đề bài. Cho ma trận embedding từ $E \in \mathbb{R}^{V\times d}$ (mỗi hàng là embedding của một từ). Ta muốn xấp xỉ hạng thấp để nén:

$$E \approx E_k, \qquad \operatorname{rank}(E_k) = k \ll d.$$

(a) Nêu cách dùng SVD $E = U\Sigma V^\top$ để tạo xấp xỉ tối ưu theo chuẩn Frobenius.
(b) Viết biểu thức sai số xấp xỉ $\|E - E_k\|_F$ theo các giá trị suy biến.
(c) Ước lượng số tham số cần lưu khi biểu diễn E_k dưới dạng tích hai ma trận $A \in \mathbb{R}^{V\times k}$ và $B \in \mathbb{R}^{k\times d}$.

Hướng dẫn và gợi ý.

- Xấp xỉ tốt nhất: lấy k giá trị suy biến lớn nhất, Σ_k, cùng các cột tương ứng U_k, V_k; khi đó $E_k = U_k\Sigma_k V_k^\top$.
- Sai số Frobenius: $\|E - E_k\|_F^2 = \sum_{i>k} \sigma_i^2$.
- Số tham số: $Vk + kd$ (so với Vd ban đầu); thảo luận điều kiện để thực sự giảm kích thước.

4.2. Bài tập 20. Căn chỉnh embedding song ngữ bằng bài toán Procrustes trực giao (Orthogonal Procrustes)

Đề bài. Căn chỉnh embedding song ngữ (cross-lingual alignment). Cho $X, Y \in \mathbb{R}^{n\times d}$ là embedding của n từ tương ứng trong hai ngôn ngữ (đã ghép cặp). Xét bài toán Procrustes trực giao:

$$\min_{R\in\mathbb{R}^{d\times d}} \|XR - Y\|_F^2 \quad \text{s.t.} \quad R^\top R = I.$$

(a) Chứng minh có thể đưa bài toán về việc cực đại hóa $\operatorname{tr}(R^\top X^\top Y)$.
(b) Nếu $X^\top Y = U\Sigma V^\top$ là SVD, hãy nêu nghiệm $R^\star$ theo U, V.

(c) Giải thích vì sao ràng buộc trực giao giúp bảo toàn hình học (cosine/độ dài) của embedding.

Hướng dẫn và gợi ý.

- Khai triển chuẩn Frobenius và dùng tính chất $\mathrm{tr}(AB) = \mathrm{tr}(BA)$.
- Nghiệm Procrustes kinh điển: $R^\star = UV^\top$.
- Trực giao bảo toàn tích vô hướng: $(\vec{a}R)\cdot(\vec{b}R) = \vec{a}\cdot\vec{b}$, nên bảo toàn cosine.

4.3. Bài tập 21. Phân rã hạng thấp cho lớp chiếu ra từ vựng

Đề bài. Trong mô hình ngôn ngữ, lớp cuối thường là phép chiếu từ không gian ẩn $\mathbb{R}^d$ sang từ vựng kích thước V với ma trận $W_{\text{out}} \in \mathbb{R}^{V\times d}$. Xét phương án thay thế $W_{\text{out}} \approx AB$ với $A \in \mathbb{R}^{V\times k}$, $B \in \mathbb{R}^{k\times d}$ (hạng thấp).
(a) So sánh số tham số và độ phức tạp tính toán giữa hai cách.
(b) Nêu cách khởi tạo A, B từ SVD của W_{out} để giảm sai số xấp xỉ ban đầu.
(c) Thảo luận rủi ro khi chọn k quá nhỏ đối với chất lượng mô hình (đặc biệt với từ hiếm).

Hướng dẫn và gợi ý.

- Số tham số: ban đầu Vd; sau phân rã: $Vk + kd$.
- Gợi ý (b): nếu $W_{\text{out}} = U\Sigma V^\top$ thì có thể lấy $A = U_k\Sigma_k^{1/2}$, $B = \Sigma_k^{1/2}V_k^\top$.
- K quá nhỏ có thể làm mất khả năng biểu diễn các hướng tinh tế; quan sát sai số ở đuôi giá trị suy biến.

4.4. Bài tập 22. Giả nghịch đảo Moore–Penrose và nghiệm chuẩn nhỏ nhất

Đề bài. Giả sử ta muốn tìm một vector cập nhật $\Delta\vec{x} \in \mathbb{R}^d$ sao cho sau lớp tuyến tính $W \in \mathbb{R}^{m\times d}$ đạt mục tiêu $\Delta\vec{y} \in \mathbb{R}^m$:

$$W\,\Delta\vec{x} = \Delta\vec{y}.$$

Trong thực tế có thể $m < d$ (vô số nghiệm) hoặc $m > d$ (không nghiệm đúng).
(a) Định nghĩa nghiệm chuẩn nhỏ nhất (minimum-norm) bằng giả nghịch đảo Moore–Penrose: $\Delta\vec{x} = W^+\Delta\vec{y}$.
(b) Nếu $W = U\Sigma V^\top$ là SVD, hãy nêu biểu thức W^+ theo U, Σ, V.
(c) Nêu cách xử lý khi có các giá trị suy biến rất nhỏ để tránh khuếch đại nhiễu (gợi ý: truncated SVD / damping).

Hướng dẫn và gợi ý.

- Nhắc lại: nghiệm chuẩn nhỏ nhất đặc biệt quan trọng khi số chiều cập nhật lớn (ví dụ prompt/prefix tuning).
- Giả nghịch đảo: $W^+ = V\Sigma^+ U^\top$, với Σ^+ đảo các phần tử khác 0 trên đường chéo.
- Thực hành: đặt ngưỡng τ và chỉ đảo các $\sigma_i > \tau$, hoặc dùng $\sigma_i/(\sigma_i^2 + \lambda)$ (damped inverse).

4.5. Bài tập 23. Chuẩn Frobenius, chuẩn hạt nhân và liên hệ với hạng

Đề bài. Cho ma trận $M \in \mathbb{R}^{p\times q}$ có các giá trị suy biến $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$.
(a) Viết định nghĩa chuẩn Frobenius $\|M\|_F$ và chuẩn hạt nhân (nuclear norm) $\|M\|_*$.
(b) Với $\sigma(M) = (5,\ 2,\ 1,\ 0, \ldots)$, hãy tính $\|M\|_F$ và $\|M\|_*$.
(c) Giải thích vì sao chuẩn hạt nhân thường được dùng như một xấp xỉ lồi cho bài toán cực tiểu hạng (rank minimization) trong nén mô hình.

Hướng dẫn và gợi ý.

- Nhắc lại: $\|M\|_F = \sqrt{\sum_i \sigma_i^2}$, còn $\|M\|_* = \sum_i \sigma_i$.
- Trong (b), chỉ cần dùng các σ_i khác 0.
- Trong (c), gợi ý: $\operatorname{rank}(M)$ không lồi và khó tối ưu; chuẩn hạt nhân là bao lồi chặt của hạng (rank) trên quả cầu chuẩn phổ.

4.6. Bài tập 24. LoRA như cập nhật hạng thấp và liên hệ với SVD

Đề bài. Trong LoRA, ta cập nhật trọng số bằng ma trận hạng thấp: $W' = W + \Delta W$, với $\Delta W = BA$, $B \in \mathbb{R}^{d_{\text{out}}\times r}$, $A \in \mathbb{R}^{r\times d_{\text{in}}}$ và $r \ll \min(d_{\text{out}}, d_{\text{in}})$.
(a) Chứng minh $\operatorname{rank}(\Delta W) \leq r$.
(b) Giả sử ta có một cập nhật đầy đủ ΔW_{full} và muốn xấp xỉ nó bằng hạng r. Nêu cách lấy xấp xỉ tốt nhất theo chuẩn Frobenius bằng SVD.
(c) Thảo luận khi nào cập nhật hạng thấp có thể đủ để đạt hiệu quả fine-tuning trong NLP (liên hệ: nội tại hạng thấp của gradient/cập nhật).

Hướng dẫn và gợi ý.

- Dùng tính chất: $\operatorname{rank}(BA) \leq \min(\operatorname{rank}(B), \operatorname{rank}(A)) \leq r$.
- Xấp xỉ tốt nhất: lấy r giá trị suy biến lớn nhất của ΔW_{full} để tạo ΔW_r.
- Thảo luận (c): nếu biến đổi cần học chủ yếu nằm trong một không gian con có số chiều nhỏ, LoRA có thể bắt được phần "quan trọng".

5 Tensor và tích chập nâng cao trong kiến trúc hiện đại

5.1. Bài tập 25. Tích ngoài và tương tác bậc hai trong mô hình hỏi–đáp

Đề bài. Trong mô hình hỏi–đáp, giả sử ta có embedding câu hỏi $\vec{q} \in \mathbb{R}^d$ và embedding đoạn văn $\vec{p} \in \mathbb{R}^d$.
(a) Xây dựng tensor bậc 2 (ma trận) $T = \vec{q} \otimes \vec{p} = \vec{q}\vec{p}^\top \in \mathbb{R}^{d \times d}$ (tích ngoài).
(b) Giải thích ý nghĩa của phần tử T_{ij} như một dạng tương tác cặp giữa chiều i của $\vec{q}$ và chiều j của $\vec{p}$.
(c) Đề xuất một cách rút gọn thông tin của T thành một vector đặc trưng $\phi(\vec{q}, \vec{p})$ để đưa vào bộ phân loại (gợi ý: lấy đường chéo, pooling, hoặc phép co tensor).

Hướng dẫn và gợi ý.

- Nhắc lại: tích ngoài tạo biểu diễn bậc hai, giàu tương tác hơn so với tích vô hướng.
- Có thể liên hệ với bilinear pooling trong thị giác/ngôn ngữ: T lưu mọi sản phẩm $q_i p_j$.
- Rút gọn (c) gợi ý: vector hóa $\mathrm{vec}(T)$ (kích thước d^2) thường quá lớn; cân nhắc chỉ lấy diagonal, hoặc dùng phép nhân với ma trận hạng thấp để nén.

5.2. Bài tập 26. Co tensor (tensor contraction) cho Multi-Head Attention và exttteinsum

Đề bài. Cho $Q, K, V \in \mathbb{R}^{B \times H \times L \times d_h}$ (batch, số head, độ dài, chiều mỗi head).
(a) Viết điểm attention $S \in \mathbb{R}^{B \times H \times L \times L}$ dưới dạng phép co tensor (tensor contraction) theo chỉ số:

$$S_{b,h,i,j} = \frac{1}{\sqrt{d_h}} \sum_{k=1}^{d_h} Q_{b,h,i,k} K_{b,h,j,k}.$$

(b) Viết biểu thức đầu ra $O \in \mathbb{R}^{B \times H \times L \times d_h}$ sau khi nhân với V:

$$O_{b,h,i,k} = \sum_{j=1}^{L} A_{b,h,i,j} V_{b,h,j,k}, \quad A = \mathrm{softmax}(S).$$

(c) Gợi ý cách hiện thực hai phép co trên bằng `einsum` (hoặc `torch.einsum`) mà không cần viết vòng lặp.

Hướng dẫn và gợi ý.

- Nhìn (a) như tích vô hướng giữa hai vector theo chiều d_h cho mỗi cặp vị trí (i, j).

- Trong `einsum`, hãy ánh xạ chỉ số chữ cái: ví dụ `'bhid,bhjd->bhij'` cho (a) và `'bhij,bhjd->bhid'` cho (b).
- Kiểm tra kích thước trung gian để tránh nhầm lẫn; đặc biệt chú ý thứ tự chỉ số khi triển khai.

5.3. Bài tập 27. phép nhân tensor theo mode n (mode-n product) cho lớp tuyến tính áp dụng trên mọi token

Đề bài. Giả sử ta có tensor biểu diễn batch câu: $X \in \mathbb{R}^{B\times L\times d}$ (batch, độ dài, chiều embedding). Một lớp tuyến tính dùng cùng trọng số cho mọi token: $W \in \mathbb{R}^{m\times d}$.
(a) Viết đầu ra $Y \in \mathbb{R}^{B\times L\times m}$ dưới dạng *mode-3 product* (nhân theo mode embedding) của tensor X với W.
(b) Chỉ ra cách biến đổi X bằng cách làm phẳng (flatten) để thực hiện phép nhân bằng một nhân ma trận duy nhất.
(c) Thảo luận ưu/nhược điểm giữa hiện thực nguyên bản theo tensor (tensor-native) và hiện thực bằng làm phẳng trong bối cảnh tối ưu hóa trên GPU.

Hướng dẫn và gợi ý.

- Mode-3 product: xem mỗi vector token $X_{b,l,:}$ được nhân bởi $W^\top$ để ra $Y_{b,l,:}$.
- Flatten (b): gộp (B,L) thành một chiều: $X' \in \mathbb{R}^{(BL)\times d}$ rồi $Y' = X'W^\top$, sau đó reshape về (B,L,m).
- Trong (c), gợi ý: flatten thường thân thiện với GEMM; tensor-native có thể rõ nghĩa nhưng phụ thuộc backend.

5.4. Bài tập 28. Trải phẳng/tái định dạng (flatten/reshape) giữa CNN và MLP: kiểm soát kích thước

Đề bài. Trong mô hình kết hợp CNN–MLP cho NLP, giả sử sau một khối tích chập ta thu được tensor đặc trưng $F \in \mathbb{R}^{B\times C\times L'}$ (batch, kênh, độ dài mới).
(a) Viết phép làm phẳng để đưa F về ma trận $Z \in \mathbb{R}^{B\times (CL')}$ trước khi vào MLP.
(b) Nếu MLP đầu tiên có trọng số $W \in \mathbb{R}^{h\times (CL')}$, hãy viết biểu thức đầu ra $H \in \mathbb{R}^{B\times h}$.
(c) Nêu một lỗi thường gặp khi flatten/reshape liên quan đến thứ tự trục (axis order) và cách kiểm tra nhanh bằng kích thước.

Hướng dẫn và gợi ý.

- Flatten theo trục (C,L'): $Z_{b,:} = \mathrm{vec}(F_{b,:,:})$.
- MLP: $H = ZW^\top + \mathbf{1}\vec{b}^\top$ (nếu có hạng tử độ lệch (bias)).
- Lỗi thường gặp: nhầm giữa `(B,C,L')` và `(B,L',C)`; hãy in kích thước tensor trước/sau và dùng ví dụ nhỏ để xác nhận thứ tự.

5.5. Bài tập 29. Tích chập nhân quả (causal) và giãn cách (dilation): trường tiếp nhận (receptive field) trên chuỗi

Đề bài. Xét phép tích chập 1D nhân quả (causal) trên chuỗi embedding $X \in \mathbb{R}^{L\times d}$ với kernel có kích thước k và hệ số giãn cách (*dilation*) r.

(a) Mô tả công thức tính đầu ra tại vị trí t sao cho chỉ phụ thuộc các vị trí $\leq t$.
(b) Với $k = 3$ và $r = 2$, hãy liệt kê các chỉ số của đầu vào tham gia vào đầu ra tại vị trí t (giả sử đủ padding).
(c) Tính độ phủ (receptive field) theo k, r khi xếp chồng L_c lớp tích chập cùng tham số (giữ nguyên k, r).

Hướng dẫn và gợi ý.

- Tích chập nhân quả (Causal conv): đầu ra tại t dùng các vị trí $t, t-r, t-2r, \ldots, t-(k-1)r$.
- Với $k = 3, r = 2$: dùng $\{t,\ t-2,\ t-4\}$.
- Phân đầu vào (Receptive field): suy luận theo số bước lùi tối đa sau nhiều lớp; có thể tính đệ quy hoặc công thức đóng tuỳ cấu hình padding/stride.

5.6. Bài tập 30. Tích chập tách theo chiều sâu (depthwise separable convolution) và đếm tham số

Đề bài. Trong CNN cho chuỗi, với đầu vào $F \in \mathbb{R}^{B\times C\times L}$ và kernel 1D kích thước k:
(a) Tính số tham số của tích chập chuẩn (standard conv) tạo ra C' kênh đầu ra.
(b) Tính số tham số của Tích chập tách theo chiều sâu gồm: (i) tích chập theo chiều sâu (mỗi kênh một kernel), (ii) tích chập điểm (pointwise conv) (1×1) trộn kênh để ra C' kênh.
(c) So sánh hai kết quả và nêu khi nào depthwise separable conv có lợi trong mô hình NLP nhẹ (lightweight NLP).

Hướng dẫn và gợi ý.

- Tính số tham số của tích chập chuẩn 1D: tham số $\approx C' \cdot C \cdot k$ (bỏ qua hạng tử độ lệch (bias)).
- Tính số tham số của Tích chập tách theo chiều sâu: $C \cdot k$; pointwise: $C' \cdot C$; tổng: $Ck + C'C$.
- Độ lợi khi k tương đối lớn và C' không quá nhỏ; thảo luận trade-off giữa tốc độ và khả năng biểu diễn.

Kết luận

Chương 5 hệ thống hóa 30 bài tập nâng cao nhằm củng cố và mở rộng các trụ cột của Đại số tuyến tính (không gian vector, biến đổi tuyến tính, trị riêng–vector riêng, SVD, tensor và tích chập) trong các bài toán cốt lõi của AI và NLP. Xuyên suốt chương là một mạch tư duy thống nhất: (i) hình học của không gian nhúng và các phép toán vector quyết định cách mã hóa quan hệ ngữ nghĩa; (ii) các phép chiếu, bình phương tối thiểu và chính quy hóa (Ridge) là ngôn ngữ chung để mô tả suy luận tuyến tính, đồng thời làm rõ vai trò của điều kiện số đối với ổn định học; (iii) phổ trị riêng và hiệp phương sai cung cấp công cụ phân tích cấu trúc dữ liệu, từ làm trắng (whitening), giảm dị hướng (anisotropy) đến phân cụm phổ theo tiêu chí độ trơn; (iv) SVD và xấp xỉ hạng thấp là nền tảng cho nén mô hình, căn chỉnh không gian embedding và các cơ chế fine-tuning hiệu quả như LoRA; (v) các phép toán tensor và tích chập (nhân quả, giãn cách, trường tiếp nhận, tách theo chiều sâu) tạo cầu nối trực tiếp giữa biểu diễn toán học và hiện thực tính toán của kiến trúc hiện đại.

Chương 6

THỰC HÀNH CÁC DỰ ÁN AI VÀ NLP THỰC TẾ

ỨNG DỤNG ĐẠI SỐ TUYẾN TÍNH

Giới thiệu 190
1 Xây dựng hệ thống nhận dạng thực thể và ẩn danh hoá thông tin . . 190
1.1. Đề bài mô tả yêu cầu 190
1.2. Quy trình tổng quát 190
Bước 1: Chuẩn hoá dữ liệu và quy ước gán nhãn 191
Bước 2: Thiết kế mô hình và đường cơ sở 192
Bước 3: Huấn luyện, điều chỉnh và kiểm soát lỗi 192
Bước 4: Ẩn danh hoá và hậu xử lý 193
Bước 5: Đánh giá và phân tích sai lỗi (error analysis) 193
2 Xây dựng hệ thống hỏi–đáp theo ngữ cảnh 194
2.1. Đề bài mô tả yêu cầu 194
2.2. Quy trình tổng quát 194
Bước 1: Chuẩn hoá và tổ chức dữ liệu hỏi–đáp 194
Bước 2: Tách mảnh (tokenization) và ánh xạ vị trí đáp án 195
Bước 3: Lựa chọn mô hình và chiến lược huấn luyện 195
Bước 4: Đánh giá và phân tích năng lực suy luận 196
Bước 5: Mở rộng theo hướng ứng dụng 196
3 Xây dựng hệ thống tóm tắt văn bản tiếng Việt 196
3.1. Đề bài mô tả yêu cầu 196
3.2. Quy trình tổng quát 196
Bước 1: Xây dựng tiêu chuẩn tóm tắt và dữ liệu 197
Bước 2: Đường cơ sở và mô hình khuyến nghị 197
Bước 3: Huấn luyện và kiểm soát hiện tượng bịa nội dung 198
Bước 4: Đánh giá và phân tích chất lượng tóm tắt 198
Bước 5: Ứng dụng và mở rộng 198
4 Xây dựng hệ thống phát hiện nội dung độc hại và thù ghét 199
4.1. Đề bài mô tả yêu cầu 199
4.2. Quy trình tổng quát 199
Bước 1: Tiền xử lý đặc thù dữ liệu mạng xã hội 199
Bước 2: đường cơ sở và mô hình khuyến nghị 200
Bước 3: Thiết lập đánh giá đúng với bài toán mất cân bằng 200
Bước 4: Giảm thiên lệch và tăng tính an toàn khi triển khai 200
Bước 5: Tích hợp vòng lặp con người (human-in-the-loop) và báo cáo vận hành 201
5 Xây dựng hệ thống RAG: truy hồi ngữ nghĩa và trả lời có dẫn chứng 201

5.1. Đề bài mô tả yêu cầu 201
5.2. Quy trình tổng quát 201
Bước 1: Xây dựng kho tri thức và chiến lược chia đoạn 202
Bước 2: Xây dựng bộ truy hồi và đường cơ sở truy hồi 202
Bước 3: Xếp hạng lại và lựa chọn bằng chứng 202
Bước 4: Tạo sinh câu trả lời có trích dẫn và kiểm soát bịa 203
Bước 5: Đánh giá toàn hệ thống và kịch bản triển khai 203
Kết luận 203

Giới thiệu

Các chương trước đã giới thiệu các khái niệm nền tảng về biểu diễn văn bản, giảm chiều, phân cụm, mô hình chủ đề và phân loại cảm xúc. Chương này mở rộng theo hướng dự án thực hành, nhằm kết nối đại số tuyến tính với các bài toán NLP thường gặp trong triển khai thực tế: nhận dạng thực thể và ẩn danh hoá, hỏi–đáp theo ngữ cảnh, tóm tắt trừu tượng, phát hiện độc hại, và hệ thống truy hồi–tạo sinh RAG.

Mỗi dự án được trình bày theo cấu trúc: mô tả yêu cầu, quy trình tổng quát và các bước gợi ý triển khai. Phần gợi ý tập trung vào mục tiêu, lựa chọn thiết kế và các điểm kiểm tra quan trọng, đủ để người học tự hiện thực, đánh giá và mở rộng.

1 Xây dựng hệ thống nhận dạng thực thể và ẩn danh hoá thông tin

1.1. Đề bài mô tả yêu cầu

Xây dựng hệ thống nhận dạng thực thể có tên (Named Entity Recognition) trên văn bản tiếng Việt và sử dụng kết quả để ẩn danh hoá thông tin nhạy cảm. Đầu vào là các đoạn văn bản (hồ sơ, đơn từ, hội thoại, thư điện tử (email), nhật ký hệ thống, v.v.). Đầu ra gồm:

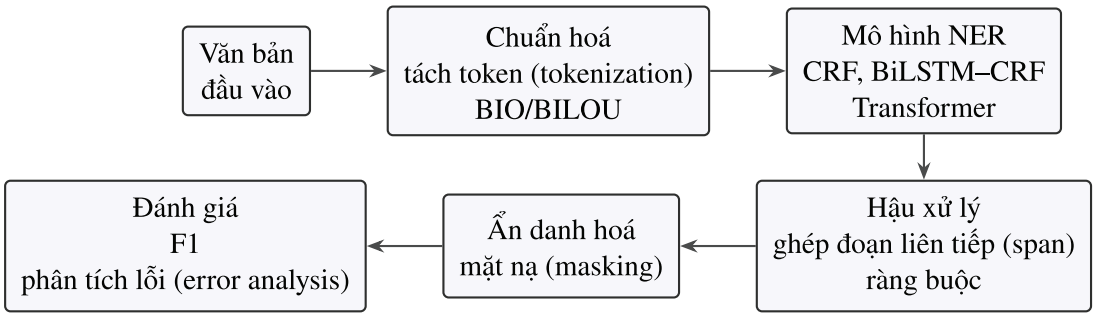
- Chuỗi nhãn theo từng mảnh (token), với các lớp như: người (`PER`), tổ chức (`ORG`), địa danh (`LOC`), thời gian (`DATE`), mã định danh (`ID`), số điện thoại (`PHONE`), thư điện tử (`EMAIL`). Có thể mở rộng thêm các lớp theo miền dữ liệu.
- Văn bản đã ẩn danh hoá theo chính sách thay thế đã định nghĩa, sử dụng các thẻ thay thế tương ứng (ví dụ: `[PER]`, `[PHONE]`, `[EMAIL]`).
- Báo cáo đánh giá gồm độ chính xác (precision), độ bao phủ (recall), F1 (F1-score), kèm phân tích sai lỗi theo lớp và theo kiểu thực thể.

1.2. Quy trình tổng quát

Chuỗi xử lý tham khảo (pipeline):

$$\text{Văn bản} \to \text{Chuẩn hoá \& Tách mảnh (tokenization)} \to \text{Gán nhãn NER} \to$$

$$\to \text{Hậu xử lý đoạn liên tiếp (span)} \to \text{Ẩn danh hoá} \to \text{Đánh giá.}$$



Hình 1: Quy trình tổng quát cho NER và ẩn danh hoá.

Bước 1: Chuẩn hoá dữ liệu và quy ước gán nhãn

Mục tiêu của bước này là thống nhất cách tách đơn vị, cách biểu diễn nhãn và chuẩn hoá văn bản, để tránh sai lệch giữa huấn luyện và suy luận.

a) Chọn sơ đồ nhãn và phạm vi thực thể

Xác định tập nhãn tối thiểu phục vụ mục tiêu ẩn danh hoá. Ưu tiên các nhãn gắn trực tiếp với thông tin nhạy cảm. Chọn một sơ đồ nhất quán như BIO (Begin–Inside–Outside) hoặc BILOU (Begin–Inside–Last–Outside–Unit). Đảm bảo quy ước về ranh giới thực thể rõ ràng cho các trường hợp nhiều từ.

b) Quy ước tách token (tokenization) và ánh xạ về nhãn

Quyết định đơn vị gán nhãn theo token mức từ. Nếu mô hình dùng subword, lưu ánh xạ subword về token gốc để tránh nhãn sai ranh giới. Thống nhất xử lý dấu câu, số, ký tự đặc biệt và từ viết tắt.

c) Chuẩn hoá văn bản theo chính sách dữ liệu

Xây dựng bộ quy tắc chuẩn hoá: khoảng trắng, dấu câu, ký hiệu lạ, lỗi gõ dấu tiếng Việt. Nếu dữ liệu có thư điện tử (email), số điện thoại, mã định danh, cần thống nhất biểu thức chính quy để nhận diện và gán nhãn nhất quán.

d) Tạo tập huấn luyện (train set), kiểm định (validation set), kiểm tra (test set) và kiểm tra rò rỉ

Chia dữ liệu theo tài liệu hoặc theo nguồn để tránh rò rỉ câu giống nhau giữa các tập. Thiết kế một kiểm tra nhanh: đếm trùng lặp chuỗi hoặc trùng lặp thực thể nhạy cảm giữa train và kiểm tra (test).

Bước 2: Thiết kế mô hình và đường cơ sở (baseline)

a) Đường cơ sở tuyến tính: trường ngẫu nhiên có điều kiện (Conditional Random Field, CRF) với đặc trưng

Thiết lập trường ngẫu nhiên có điều kiện (Conditional Random Field, CRF) như một chuẩn so sánh. Chọn nhóm đặc trưng tối thiểu: n-gram ký tự, tiền tố/hậu tố, kiểu chữ, mẫu số, dấu câu. Mục tiêu là tạo một mốc F1 để so với mô hình sâu.

b) Đường cơ sở sâu: BiLSTM–trường ngẫu nhiên có điều kiện (Conditional Random Field, CRF)

Tổ chức biểu diễn embedding mức từ hoặc mức ký tự, đưa qua BiLSTM và tầng trường ngẫu nhiên có điều kiện (Conditional Random Field, CRF). Chú ý mặt nạ độ dài chuỗi và cách xử lý từ ngoài từ vựng.

c) Mô hình khuyến nghị: kiến trúc Transformer cho token classification

Sử dụng mô hình PhoBERT hoặc một kiến trúc Transformer tiếng Việt khác, gắn đầu ra (head) phân loại token. Nếu dùng trường ngẫu nhiên có điều kiện phía trên kiến trúc Transformer, cần kiểm tra tính ổn định và chi phí huấn luyện.

d) Tiêu chí chọn mô hình

Chọn mô hình theo cân bằng giữa F1, tốc độ suy luận, độ dễ kiểm soát lỗi và khả năng tổng quát trên miền dữ liệu mới. Trong bối cảnh ẩn danh hoá, ưu tiên độ bao phủ (recall) cao cho nhãn nhạy cảm, nhưng cần kiểm soát false positive để tránh làm biến dạng văn bản.

Bước 3: Huấn luyện, điều chỉnh và kiểm soát lỗi

a) Thiết lập thực nghiệm tối thiểu

Cố định bộ siêu tham số cơ bản: batch size, learning rate, số epoch, weight decay và seed. Theo dõi F1 trên kiểm định (validation) theo từng nhãn.

b) Chiến lược xử lý mất cân bằng nhãn

Nếu nhãn hiếm, ưu tiên tăng dữ liệu có kiểm soát hoặc chiến lược lấy mẫu. Tránh làm méo phân phối câu bằng cách nhân bản quá mức các mẫu ngắn.

c) Kiểm soát quá khớp (overfitting) và chọn mốc lưu mô hình (checkpoint)

Dùng dừng sớm (early stopping) dựa trên F1 tổng và F1 cho nhóm nhãn nhạy cảm. Lưu kèm cấu hình tách mảnh (tokenization) và chuẩn hoá để suy luận đồng nhất.

d) Bộ kiểm tra lỗi thường gặp

Chuẩn bị một danh sách câu kiểm thử gồm các mẫu số điện thoại, thư điện tử (email), mã số, họ tên, địa chỉ. Mỗi lần thay đổi tiền xử lý hoặc tách token (tokenization), chạy lại bộ kiểm tra để phát hiện lỗi ranh giới đoạn liên tiếp (span).

Bước 4: Ẩn danh hoá và hậu xử lý

a) Ghép đoạn liên tiếp từ chuỗi nhãn

Từ chuỗi BIO/BILOU, ghép các đoạn liên tiếp (span) liên tục và xác định loại thực thể. Cài đặt ràng buộc tối thiểu: `I-X` chỉ hợp lệ khi trước đó có `B-X` hoặc `I-X`.

b) Chính sách thay thế và bảo toàn định dạng

Định nghĩa ánh xạ `TYPE` $\mapsto$ mẫu thay thế, ví dụ `[PER]`, `[PHONE]`. Nếu cần bảo toàn định dạng, có thể thay số bằng mẫu giữ độ dài, hoặc thay thế theo nhóm quy tắc.

c) Kết hợp luật và mô hình

Với các trường có cấu trúc mạnh (thư điện tử (email), số điện thoại, mã số), ưu tiên kết hợp biểu thức chính quy để tăng độ phủ. Thiết kế thứ tự áp dụng: luật trước hay mô hình trước, và quy ước xử lý khi xung đột.

d) Kiểm thử hồi quy cho ẩn danh hoá

Tạo bộ kiểm thử đầu vào–đầu ra. Mỗi mẫu có văn bản gốc, nhãn kỳ vọng và văn bản sau thay thế. Mục tiêu là đảm bảo hệ thống ổn định khi cập nhật mô hình hoặc thay đổi chính sách.

Bước 5: Đánh giá và phân tích sai lỗi (error analysis)

a) Định nghĩa độ đo ở mức đoạn liên tiếp (span)

Báo cáo độ chính xác (precision), độ bao phủ (recall), F1 ở mức đoạn liên tiếp (span). Nêu rõ tiêu chuẩn khớp: khớp chính xác ranh giới hay cho phép khớp một phần.

b) Phân tích sai lỗi theo lớp và theo dạng

Lập bảng nhầm lẫn giữa các nhãn chính. Nhóm lỗi theo dạng: sai ranh giới, nhầm loại, bỏ sót, gán thừa. Chọn 20 ví dụ điển hình để thảo luận.

c) Đánh giá theo mục tiêu ẩn danh hoá

Tách báo cáo cho nhóm nhãn nhạy cảm. Nếu mục tiêu là an toàn dữ liệu, ưu tiên chỉ số độ bao phủ (recall) và tỷ lệ bỏ sót. Đồng thời báo cáo tỷ lệ biến dạng văn bản do gán nhãn thừa.

d) Khuyến nghị cải tiến

Từ error analysis, đề xuất một cải tiến mỗi vòng lặp: mở rộng dữ liệu cho trường hợp khó, điều chỉnh tách mảnh (tokenization), thêm luật hậu xử lý, hoặc thay đổi chính sách nhãn.

2 Xây dựng hệ thống hỏi–đáp theo ngữ cảnh

2.1. Đề bài mô tả yêu cầu

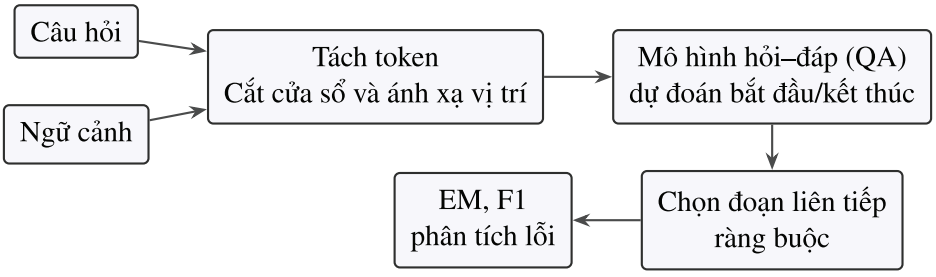
Cho một tập văn bản nguồn, hệ thống cần trả lời câu hỏi dựa trên ngữ cảnh cho trước. Xét biến thể hỏi–đáp trích xuất: dự đoán vị trí bắt đầu và kết thúc của đáp án trong văn bản. Đầu ra gồm đáp án trích xuất, điểm tin cậy và báo cáo đánh giá theo khớp chính xác (exact match, EM) và F1 theo mức từ (token-level).

2.2. Quy trình tổng quát

Chuỗi xử lý tham khảo (pipeline):

$$(\text{Câu hỏi},\ \text{Ngữ cảnh}) \to \text{Tách token (tokenization) và Cắt cửa sổ}$$

$$\to \text{Mô hình QA} \to \text{Chọn đoạn liên tiếp (span)} \to \text{Đánh giá}.$$



Hình 2: Quy trình tổng quát cho hỏi–đáp trích xuất.

Bước 1: Chuẩn hoá và tổ chức dữ liệu hỏi–đáp

a) Xác định dạng dữ liệu và trường bắt buộc

Mỗi mẫu cần có `context`, `question`, `answer_text` và `answer_start`. Nếu có câu hỏi không có đáp án, bổ sung nhãn `is_impossible`.

b) Chuẩn hoá ngữ cảnh và câu hỏi

Thống nhất chuẩn hoá dấu câu, khoảng trắng và ký tự lạ. Tạo một hàm chuẩn hoá duy nhất dùng cho cả train và inference.

c) Chia tập dữ liệu để tránh rò rỉ

Tách theo bài viết hoặc theo tài liệu nguồn, tránh việc cùng một đoạn ngữ cảnh xuất hiện ở nhiều tập. Báo cáo thống kê độ dài ngữ cảnh và độ dài đáp án để định hướng cắt cửa sổ.

Bước 2: Tách token và ánh xạ vị trí đáp án

a) Tách token theo mô hình và lưu ánh xạ ký tự

Tách token dựa vào công cụ tokenizer của mô hình. Lưu ánh xạ giữa chỉ số ký tự gốc và chỉ số token để truy hồi ranh giới đáp án.

b) Cắt cửa sổ ngữ cảnh dài

Nếu `context` dài, dùng cửa sổ trượt (sliding window) với bước trượt phù hợp. Mục tiêu là đảm bảo mọi đáp án đều nằm trọn trong ít nhất một cửa sổ.

c) Kiểm tra lại nhãn đáp án sau chuẩn hoá

Sau khi chuẩn hoá, xác nhận `answer_text` khớp lại với chuỗi con bắt đầu tại `answer_start`. Nếu lệch, ưu tiên sửa ở mức dữ liệu trước khi huấn luyện.

Bước 3: Lựa chọn mô hình và chiến lược huấn luyện

a) Đường cơ sở và mô hình khuyến nghị

Đường cơ sở: mô hình QA dựa trên kiến trúc Transformer tiếng Việt với đầu ra dự đoán bắt đầu/kết thúc (start/end). Mô hình khuyến nghị: chúng ta có thể tinh chỉnh (fine-tune) PhoBERT hoặc một mô hình nền mạnh hơn nếu dữ liệu lớn.

b) Ràng buộc chọn đoạn liên tiếp

Khi suy luận, chọn cặp (start, end) thoả start $\leq$ end và độ dài không vượt ngưỡng. Có thể dùng top-k start và top-k end để giảm chi phí.

c) Huấn luyện ổn định

Theo dõi hàm mất mát (loss) và EM/F1 trên kiểm định (validation). Chọn mốc lưu mô hình (checkpoint) theo F1. Nếu có câu hỏi không đáp án, bổ sung ngưỡng quyết định hoặc đầu ra phân loại.

Bước 4: Đánh giá và phân tích năng lực suy luận

a) Chuẩn hoá văn bản trước khi chấm

EM và F1 thường dựa trên chuẩn hoá: chữ thường, bỏ dấu câu, chuẩn hoá khoảng trắng. Công bố rõ chuẩn hoá để tái lập kết quả.

b) Phân tích lỗi theo loại câu hỏi

Nhóm câu hỏi theo dạng: ai, ở đâu, khi nào, định nghĩa, số liệu. Thống kê F1 theo nhóm để biết mô hình yếu ở đâu.

c) Kiểm tra tính bền vững theo độ dài

So sánh kết quả theo độ dài ngữ cảnh và độ dài đáp án. Nếu suy giảm mạnh khi ngữ cảnh dài, ưu tiên cải thiện chiến lược cắt cửa sổ hoặc dùng mô hình dài ngữ cảnh.

Bước 5: Mở rộng theo hướng ứng dụng

a) Tích hợp truy hồi ngữ cảnh

Với kho tài liệu lớn, thêm bước truy hồi trước QA. Thiết kế: bộ truy hồi chọn top-k đoạn, sau đó QA trên từng đoạn và hợp nhất.

b) Thiết kế API và logging

Định nghĩa giao diện: `question`, `context_id`, đáp án và độ tin cậy. Ghi log câu hỏi khó để mở rộng dữ liệu và cải thiện mô hình theo vòng lặp.

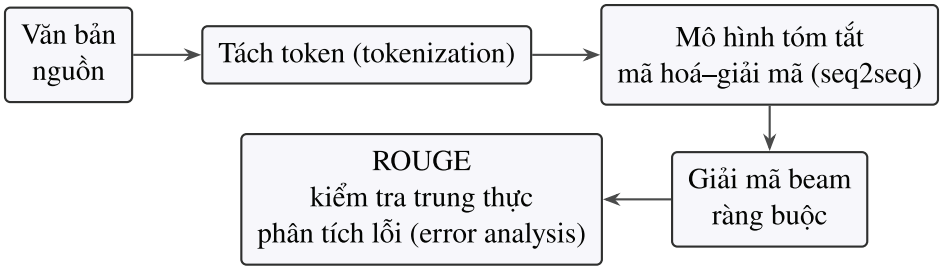
3 Xây dựng hệ thống tóm tắt văn bản tiếng Việt

3.1. Đề bài mô tả yêu cầu

Xây dựng hệ thống tạo tóm tắt trừu tượng cho văn bản tiếng Việt (tin tức, báo cáo, văn bản hành chính). Đầu vào là văn bản nguồn và đầu ra là bản tóm tắt ngắn, đúng trọng tâm và nhất quán với nội dung nguồn. Báo cáo kết quả bằng ROUGE và phân tích chất lượng theo tiêu chí nội dung, mạch lạc và độ trung thực.

3.2. Quy trình tổng quát

$$\text{Văn bản nguồn} \to \text{Tiền xử lý} \to \text{Mô hình mã hoá–giải mã (seq2seq)} \to$$
$$\to \text{Giải mã} \to \text{Đánh giá \& kiểm soát độ trung thực.}$$



Hình 3: Quy trình tổng quát cho tóm tắt văn bản.

Bước 1: Xây dựng tiêu chuẩn tóm tắt và dữ liệu

a) Định nghĩa độ dài và phong cách tóm tắt

Xác định số câu hoặc số token cho bản tóm tắt. Quy định phong cách: một đoạn văn hay gạch đầu dòng, mức độ trang trọng, có giữ tên riêng hay không.

b) Chuẩn hoá dữ liệu và tách tập

Lọc văn bản quá ngắn hoặc quá dài. Chuẩn hoá dấu câu và khoảng trắng. Chia tập theo nguồn để tránh trùng lặp bài viết.

c) Tạo đường cơ sở đơn giản

Đường cơ sở trích xuất: lấy câu đầu hoặc theo TF–IDF để có mốc ROUGE, giúp kiểm tra dữ liệu và quy trình chấm.

Bước 2: đường cơ sở và mô hình khuyến nghị

a) Đường cơ sở mã hoá–giải mã (seq2seq)

Chọn một mô hình encoder–decoder phù hợp tiếng Việt hoặc đa ngữ. Thiết lập tinh chỉnh (fine-tune) với giới hạn độ dài đầu vào và đầu ra rõ ràng.

b) Điều kiện hoá theo miền dữ liệu

Nếu có nhiều miền (tin tức, báo cáo, hành chính), cân nhắc thêm thẻ miền ở đầu vào. Mục tiêu là ổn định phong cách và thuật ngữ.

c) Chiến lược giải mã

Ưu tiên tìm kiếm chùm (beam search) với độ rộng vừa phải. Thiết kế một ràng buộc lặp lại n-gram để giảm lặp câu.

Bước 3: Huấn luyện và kiểm soát hiện tượng bịa nội dung

a) Thiết lập huấn luyện tối thiểu

Theo dõi hàm mất mát (loss) và ROUGE trên kiểm định (validation). Chọn mốc lưu mô hình (checkpoint) theo ROUGE-L hoặc theo tiêu chí nội dung.

b) Kiểm tra độ trung thực

Chọn một tập nhỏ và kiểm tra thủ công: các số liệu, tên riêng, mốc thời gian trong tóm tắt có xuất hiện trong nguồn hay không. Báo cáo tỷ lệ sai lệch và ví dụ minh hoạ.

c) Gợi ý biện pháp giảm bịa

Giảm độ tự do giải mã bằng ràng buộc lặp lại, giảm độ rộng beam, hoặc tăng trọng số phạt độ dài. Nếu dữ liệu cho phép, cân nhắc học ràng buộc dựa trên trích dẫn đoạn liên tiếp quan trọng.

Bước 4: Đánh giá và phân tích chất lượng tóm tắt

a) Đánh giá tự động bằng ROUGE

Báo cáo ROUGE-1, ROUGE-2, ROUGE-L. Nêu rõ cách tách từ và chuẩn hoá tiếng Việt khi tính ROUGE.

b) Đánh giá theo tiêu chí nội dung và mạch lạc

Thiết kế một thang điểm đơn giản cho ba tiêu chí: bao phủ ý chính, mạch lạc, độ trung thực. Chấm trên một tập nhỏ để bổ trợ cho ROUGE.

c) Phân tích lỗi

Nhóm lỗi: lặp, bỏ sót ý chính, thêm chi tiết không có trong nguồn, sai số liệu, sai tên riêng. Đề xuất một cải tiến cho mỗi nhóm lỗi.

Bước 5: Ứng dụng và mở rộng

a) Tóm tắt dài ngữ cảnh

Nếu văn bản dài, dùng chiến lược chia đoạn và tóm tắt theo cấp. Thiết kế: tóm tắt từng đoạn, rồi tóm tắt lại các tóm tắt.

b) Tích hợp với truy hồi và trích dẫn

Trong môi trường yêu cầu dẫn chứng, kết hợp bước truy hồi các đoạn liên quan và yêu cầu mô hình tạo tóm tắt có tham chiếu.

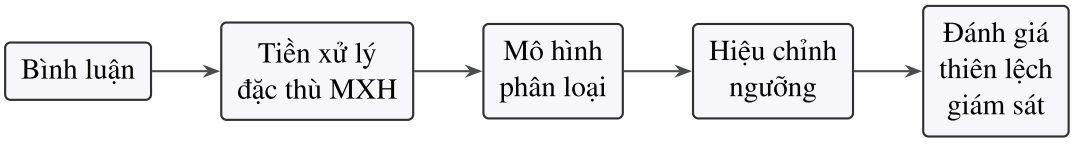
4 Xây dựng hệ thống phát hiện nội dung độc hại và thù ghét

4.1. Đề bài mô tả yêu cầu

Xây dựng bộ phân loại phát hiện nội dung độc hại, công kích hoặc thù ghét trong bình luận tiếng Việt. Đầu vào là câu hoặc đoạn ngắn, đầu ra là nhãn (độc hại, thù ghét, bình thường hoặc đa nhãn) kèm điểm tin cậy. Cần báo cáo đánh giá trong điều kiện mất cân bằng dữ liệu và phân tích thiên lệch.

4.2. Quy trình tổng quát

Bình luận → Tiền xử lý → Mô hình phân loại → Hiệu chỉnh ngưỡng → Đánh giá & giám sát.



Hình 4: Quy trình tổng quát cho phát hiện độc hại.

Bước 1: Tiền xử lý đặc thù dữ liệu mạng xã hội

a) Chuẩn hoá chính tả và ký hiệu

Chuẩn hoá viết tắt, lặp ký tự, emoticon, hashtag và dấu câu. Thiết kế một danh sách thay thế tối thiểu để giảm nhiễu nhưng không làm mất tín hiệu độc hại.

b) Xử lý từ nhạy cảm và biến thể né tránh

Thu thập biến thể viết sai hoặc cố tình che từ nhạy cảm. Cân nhắc chuẩn hoá theo mẫu hoặc bổ sung đặc trưng ký tự để mô hình nhận biết.

c) Kiểm tra phân phối độ dài và tỷ lệ ký tự đặc biệt

Báo cáo thống kê giúp phát hiện dữ liệu bất thường và hỗ trợ quyết định cắt độ dài.

Bước 2: Đường cơ sở và mô hình khuyến nghị

a) Đường cơ sở tuyến tính

TF–IDF kết hợp Logistic Regression hoặc SVM là đường cơ sở mạnh cho văn bản ngắn. Mục tiêu là có mốc so sánh và kiểm tra pipeline.

b) Mô hình kiến trúc Transformer

Chúng ta có thể tinh chỉnh PhoBERT cho phân loại câu. Nếu bài toán đa nhãn, dùng sigmoid và ngưỡng theo từng nhãn.

c) Hiệu chỉnh xác suất

Dùng hiệu chuẩn xác suất (calibration) trên tập kiểm định (validation) để điểm tin cậy có ý nghĩa khi triển khai.

Bước 3: Thiết lập đánh giá đúng với bài toán mất cân bằng

a) Chọn độ đo phù hợp

Báo cáo macro F1 và micro F1. Nếu nhãn hiếm, ưu tiên macro F1 và PR-AUC theo nhãn.

b) Chia tập và kiểm tra rò rỉ

Tách theo người dùng hoặc theo chủ đề nếu có, để tránh rò rỉ mẫu gần trùng. Báo cáo độ ổn định qua nhiều seed.

c) Phân tích ngưỡng quyết định

Vẽ đường PR và chọn ngưỡng theo mục tiêu vận hành: giảm bỏ sót hay giảm cảnh báo sai.

Bước 4: Giảm thiên lệch và tăng tính an toàn khi triển khai

a) Kiểm tra thiên lệch theo nhóm từ khoá

Tạo các nhóm từ khoá nhạy cảm (giới, vùng miền, nghề nghiệp) và kiểm tra tỷ lệ gán nhãn độc hại. Mục tiêu là phát hiện thiên lệch do dữ liệu.

b) Thiết kế cơ chế cảnh báo an toàn

Nếu điểm tin cậy thấp hoặc câu mơ hồ, chuyển sang trạng thái cần kiểm duyệt thay vì tự động kết luận. Xác định rõ quy tắc fallback.

c) Ghi nhận dữ liệu mới và drift

Thiết kế logging tối thiểu: phân phối điểm, tỷ lệ nhãn, top từ khoá. Khi drift, ưu tiên thu thập dữ liệu mới và tinh chỉnh theo vòng lặp.

Bước 5: Giao diện phản hồi và báo cáo vận hành

a) Giao diện phản hồi và nhãn hoá lại

Thiết kế quy trình để người kiểm duyệt phản hồi và cập nhật nhãn. Ưu tiên lưu trường hợp mô hình sai tự tin.

b) Bộ chỉ số vận hành

Theo dõi tỷ lệ cảnh báo, tỷ lệ đảo nhãn bởi con người, thời gian xử lý, và tỷ lệ khiếu nại. Các chỉ số này bổ trợ cho F1 trong môi trường thực.

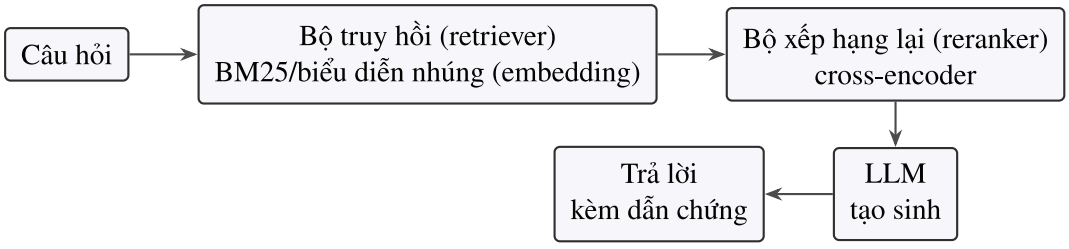
5 Xây dựng hệ thống RAG: truy hồi ngữ nghĩa và trả lời có dẫn chứng

5.1. Đề bài mô tả yêu cầu

Xây dựng hệ thống trả lời câu hỏi dựa trên kho tri thức nội bộ hoặc tài liệu công khai. Hệ thống cần truy hồi các đoạn liên quan và tạo câu trả lời kèm dẫn chứng (các đoạn trích hoặc mã tài liệu). Đầu ra gồm: câu trả lời, danh sách bằng chứng, và báo cáo đánh giá chất lượng truy hồi và chất lượng trả lời.

5.2. Quy trình tổng quát

$$\text{Câu hỏi} \to \text{Bộ truy hồi (retriever)} \to \text{Bộ xếp hạng lại (reranker)} \to$$
$$\to \text{Bộ sinh (generator)} \to \text{Câu trả lời có trích dẫn.}$$



Hình 5: Quy trình tổng quát cho RAG.

Bước 1: Xây dựng kho tri thức và chiến lược chia đoạn

a) Thu thập và chuẩn hoá tài liệu

Xác định phạm vi tài liệu và loại trừ phiên bản lỗi thời. Chuẩn hoá encoding, loại bỏ nội dung trùng lặp và tách cấu trúc (tiêu đề, mục, bảng).

b) Chia đoạn và gắn metadata

Chọn kích thước đoạn theo số token, có chồng lấn để giữ ngữ cảnh. Gắn metadata: nguồn, tiêu đề, mục, thời gian, đường dẫn, để phục vụ trích dẫn.

c) Thiết kế chỉ mục

Tạo chỉ mục từ khoá (BM25) và chỉ mục vector cho truy hồi ngữ nghĩa. Thiết lập một tập câu hỏi thử để kiểm tra nhanh chất lượng truy hồi.

Bước 2: Xây dựng bộ truy hồi và đường cơ sở truy hồi

a) Đường cơ sở từ khoá

Dùng BM25 làm đường cơ sở để có mốc so sánh. Báo cáo độ bao phủ (recall)@k cho tập câu hỏi chuẩn.

b) Bộ truy hồi ngữ nghĩa

Dùng biểu diễn nhúng (embedding) cho câu hỏi và đoạn, truy hồi bằng cosine hoặc dot product. Thiết kế chiến lược kết hợp: hợp nhất danh sách top-k từ BM25 và vector.

c) Kiểm tra nhiễu và trùng lặp

Đảm bảo top-k không bị trùng đoạn và không chứa nhiều đoạn rất giống nhau. Nếu trùng lặp cao, điều chỉnh chia đoạn hoặc lọc theo độ tương tự.

Bước 3: Xếp hạng lại (reranking) và lựa chọn bằng chứng

a) Bộ xếp hạng lại (reranker) dạng cross-encoder

Huấn luyện hoặc dùng sẵn một cross-encoder để sắp hạng lại top-n. Mục tiêu là tăng độ chính xác (precision) của bằng chứng.

b) Quy tắc lựa chọn bằng chứng

Chọn số đoạn bằng chứng vừa đủ, ưu tiên độ đa dạng về nguồn và mục. Nếu trả lời cần nhiều bước, giữ các đoạn bổ trợ thay vì chỉ đoạn có điểm cao nhất.

c) Định dạng trích dẫn

Chuẩn hoá cách trích dẫn: mã đoạn, tiêu đề, vị trí trong tài liệu. Thiết kế để người dùng có thể truy vết nhanh nguồn.

Bước 4: Tạo sinh câu trả lời có trích dẫn và kiểm soát bịa

a) Thiết kế lời nhắc (prompt) ràng buộc theo bằng chứng

Đưa câu hỏi và các đoạn bằng chứng vào lời nhắc (prompt), yêu cầu mô hình chỉ dựa trên bằng chứng. Quy định định dạng đầu ra: câu trả lời và danh sách trích dẫn.

b) Kiểm soát bịa bằng hậu kiểm

Sau khi sinh, kiểm tra câu trả lời có chứa thông tin không xuất hiện trong bằng chứng hay không, đặc biệt với số liệu và tên riêng. Thiết kế một bộ kiểm tra đơn giản dựa trên so khớp từ khoá và quy tắc.

c) Chiến lược khi bằng chứng yếu

Nếu bộ truy hồi không tìm được bằng chứng đủ mạnh, hệ thống trả lời không chắc chắn và đề nghị người dùng cung cấp thêm ngữ cảnh, thay vì suy đoán.

Bước 5: Đánh giá toàn hệ thống và kịch bản triển khai

a) Đánh giá truy hồi

Báo cáo độ bao phủ (recall)@k, MRR hoặc nDCG trên tập câu hỏi có nhãn bằng chứng. Nếu không có nhãn, dùng đánh giá thủ công có hướng dẫn.

b) Đánh giá câu trả lời

Thiết kế tiêu chí: đúng nội dung, có dẫn chứng phù hợp, và không bịa. Chấm trên một tập nhỏ để bổ trợ cho độ đo tự động.

c) Triển khai và giám sát

Thiết kế caching, giới hạn độ dài context, và logging truy vấn. Theo dõi drift của kho tri thức và lịch cập nhật chỉ mục.

Kết luận

Chương 6 đã hệ thống hoá các dự án thực hành tiêu biểu trong trí tuệ nhân tạo và xử lý ngôn ngữ tự nhiên với mục tiêu giúp người học từ mức hiểu mô hình đến

mức xây hệ thống đối với dữ liệu và yêu cầu triển khai thực tế. Các bài toán đại diện bao gồm nhận dạng thực thể có tên (Named Entity Recognition, NER) phục vụ ẩn danh hoá (de-identification), hỏi–đáp trích xuất (extractive question answering), tóm tắt văn bản (text summarization), phân loại cảm xúc/độc hại trên mạng xã hội (sentiment/toxicity classification), và truy hồi–tạo sinh tăng cường (retrieval-augmented generation, RAG).

Về phương pháp luận, mỗi dự án được chuẩn hoá theo tuyến quy trình *dữ liệu–mô hình–huấn luyện–đánh giá–vận hành*: (i) chuẩn hoá dữ liệu và quy ước gán nhãn nhằm giảm nhiễu và hạn chế rò rỉ; (ii) xây dựng đường cơ sở để thiết lập mốc so sánh khách quan; (iii) lựa chọn mô hình khuyến nghị theo đặc tính bài toán và ràng buộc tài nguyên; (iv) thiết kế thước đo, báo cáo và phân tích sai lỗi (error analysis) để nhận diện nguyên nhân gốc rễ; và (v) bổ sung các cơ chế triển khai như hiệu chuẩn xác suất (calibration), giám sát chất lượng, cùng vòng lặp khi dữ liệu/nhãn cần được cập nhật liên tục.

Về mặt kỹ thuật, đại số tuyến tính đóng vai trò trụ cột trong toàn bộ chuỗi xử lý: biểu diễn ngôn ngữ dưới dạng véc-tơ (vector) thông qua biểu diễn nhúng (embedding), ước lượng mức độ liên quan bằng độ tương tự (similarity) như cosine, tối ưu hoá dựa trên đạo hàm và cập nhật tham số trong không gian nhiều chiều, cũng như các phép biến đổi tuyến tính trong các lớp mạng và cơ chế chú ý (attention) dựa trên nhân ma trận. Trong hệ thống RAG, việc thiết kế bộ truy hồi (retriever) và bộ xếp hạng lại (reranker), chuẩn hoá truy vấn–tài liệu trong không gian véc-tơ, và ràng buộc tạo sinh theo bằng chứng là các điểm quyết định để nâng cao tính đúng đắn và độ tin cậy của câu trả lời.

Tổng kết lại, chương 6 cung cấp một khung thực hành có tính hệ thống, hàn lâm và có thể tái lập cho các dự án AI/NLP, giúp người học không chỉ đạt chất lượng mô hình, mà còn kiểm soát được rủi ro triển khai (sai lệch dữ liệu, bịa nội dung, thiên lệch, và an toàn). Trên nền tảng này, có thể mở rộng theo các hướng nâng cao như tối ưu truy hồi đa ngôn ngữ, học từ phản hồi của người dùng, tích hợp tri thức có cấu trúc, và xây dựng quy trình đánh giá phù hợp cho các hệ tạo sinh trong môi trường vận hành thực.

Một số tài nguyên dữ liệu tham khảo

- VLSP NER: `https://vlsp.org.vn/resources`
- PhoNER_COVID19: `https://github.com/VinAIResearch/PhoNER`$_{COVID19}$
- `UIT-ViQuAD: https://arxiv.org/abs/2009.14725 (kèm liên kết dữ liệu trong bài)`
- `ViHSD: https://github.com/sonlam1102/vihsd`

- UIT-ViCTSD: https://github.com/tarudesu/ViCTSD
- BEIR: https://github.com/beir-cellar/beir
- mMARCO: https://github.com/unicamp-dl/mMARCO

KẾT LUẬN VÀ TỔNG KẾT

Cuốn sách *Đại số tuyến tính và ứng dụng trong Trí tuệ nhân tạo và Xử lý ngôn ngữ tự nhiên* được biên soạn với mục tiêu xuyên suốt: giúp bạn đọc nhìn thấy đại số tuyến tính không chỉ là một hệ thống ký hiệu và phép biến đổi, mà là ngôn ngữ nền tảng của dữ liệu và mô hình trong AI và NLP. Trên tinh thần đó, nội dung sách được tổ chức theo lộ trình từ nền tảng đến thực hành, từ khái niệm đến tác vụ, và từ tác vụ đến triển khai dự án, nhằm hình thành cho người học một năng lực hiểu đúng, làm được và giải thích được.

Trước hết, các chương đầu đặt nền móng bằng việc hệ thống hóa các cấu trúc trung tâm của đại số tuyến tính (vector, ma trận, tensor) và các phép toán then chốt đi kèm. Trên nền tảng ấy, chương bài tập được thiết kế theo hướng gắn chặt với dữ liệu ngôn ngữ tự nhiên, giúp bạn đọc thực hành các kỹ thuật biểu diễn văn bản (one-hot, Bag-of-Words, TF–IDF, embedding), các phép toán đo tương đồng và biến đổi tuyến tính, cũng như các công cụ phân rã (eigen, SVD/LSA) và những phép toán đặc trưng trong học sâu (tích chập, attention, thao tác trên tensor). Cách tiếp cận bài tập – lời giải – nhận xét và liên hệ NLP – hình minh họa – mã Python được duy trì nhất quán để bảo đảm tính sư phạm, tính chặt chẽ và khả năng tái lập.

Tiếp theo, các chương dự án đóng vai trò kết nối lý thuyết với thực tiễn: bạn đọc được hướng dẫn triển khai các pipeline trong NLP như tìm kiếm tài liệu tương đồng dựa trên không gian vector và độ đo tương đồng, phân cụm văn bản trên embedding kết hợp giảm chiều, cũng như xác định chủ đề và trực quan hóa không gian ngữ nghĩa dựa trên LSA/SVD. Trọng tâm của phần này là giúp người học tư duy mô hình hóa: biết dữ liệu đang nằm ở đâu trong không gian, phép toán nào đang chi phối tín hiệu và vì sao kết quả lại hợp lý về mặt hình học và đại số.

Đặc biệt, Chương 4 vừa là một chương ứng dụng nâng cao, vừa là một cầu nối có chủ đích của tác giả sang các miền toán học thiết yếu khác trong AI và NLP. Thông qua dự án phân tích cảm xúc sử dụng TF–IDF và Logistic Regression, bạn đọc được thấy rằng để mô tả, huấn luyện và đánh giá một mô hình học máy, đại số tuyến tính là chưa đủ: ta cần ngôn ngữ xác suất để diễn giải đầu ra như một xác suất có điều kiện, và cần giải tích và tối ưu để xây dựng hàm mất mát, tính gradient và cập nhật tham số. Dụng ý của chương này là giúp bạn đọc nhận ra tính liên ngành nội tại của toán học trong AI: các mảnh ghép đại số tuyến tính, xác suất và giải tích không đứng tách rời, mà gắn kết chặt chẽ trong mọi pipeline hiện đại.

Từ tổng thể nội dung, có thể khẳng định rằng việc học đại số tuyến tính theo hướng ứng dụng không chỉ giúp người học biết công thức, mà quan trọng hơn là hình thành năng lực mô hình hóa và suy luận trong không gian véc-tơ: hiểu dữ liệu dưới dạng cấu trúc, hiểu thuật toán dưới dạng phép biến đổi, và hiểu mô hình dưới dạng một chuỗi ánh xạ có thể kiểm chứng bằng tính toán và trực quan hóa. Đây là nền tảng bền vững để bạn đọc tiếp tục tiếp cận các mô hình hiện đại hơn một cách vững vàng,

có hệ thống và có khả năng giải thích.

Cuối cùng, tác giả xin bày tỏ mong muốn cuốn sách này không chỉ là một tài liệu tham khảo, mà còn là một bản đồ học tập để bạn đọc tự tin triển khai các bài toán AI và NLP từ cơ bản đến nâng cao. Rất mong bạn đọc tiếp tục đón đọc các cuốn sách tiếp theo của Bộ sách *Toán ứng dụng trong Trí tuệ nhân tạo và Xử lý ngôn ngữ tự nhiên* sắp xuất bản gồm:

1. Xác suất Thống kê và ứng dụng trong Trí tuệ nhân tạo và Xử lý ngôn ngữ tự nhiên.
2. Giải tích và ứng dụng trong Trí tuệ nhân tạo và Xử lý ngôn ngữ tự nhiên.

để khám phá sâu hơn nền tảng toán học và tính ứng dụng kỳ diệu của toán học trong thế giới AI và NLP.

Tài liệu tham khảo

[1] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In: Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), pp. 4171–4186 (2019).

[2] Eisenstein, J.: *Introduction to Natural Language Processing*. The MIT Press, Cambridge, MA (2019).

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., Polosukhin, I.: *Attention Is All You Need*. In: Advances in Neural Information Processing Systems 30 (NeurIPS 2017), pp. 5998–6008 (2017).

[4] Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. The MIT Press, Cambridge, MA (2016).

[5] Strang, G.: *Introduction to Linear Algebra*. 5th edn. Wellesley-Cambridge Press, Wellesley, MA (2016).

[6] Axler, S.: *Linear Algebra Done Right*. 3rd edn. Springer, Cham (2015).

[7] Mikolov, T., Chen, K., Corrado, G., Dean, J.: *Efficient Estimation of Word Representations in Vector Space*. arXiv preprint arXiv:1301.3781 (2013).

[8] Huynh, C.-P.: *Système d'exploitation de corpus de traductions: Des suites de test pour la TA à un système d'exploitation de corpus alignés de documents et métadocuments multilingues, multiannotés et multimédia*. Éditions universitaires européennes, Saarbrücken (2011).

[9] Manning, C. D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, Cambridge (2008).

ĐẠI SỐ TUYẾN TÍNH VÀ ỨNG DỤNG
TRONG TRÍ TUỆ NHÂN TẠO VÀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

(Sách chuyên khảo)

Chịu trách nhiệm xuất bản, nội dung:
Giám đốc - Tổng biên tập: TS. TRẦN CHÍ ĐẠT
Chịu trách nhiệm bản thảo:
Phó Giám đốc - Phó Tổng biên tập: NGÔ THỊ MỸ HẠNH

Biên tập nội dung	: Nguyễn Tiến Sỹ Lê Hồ Diệu Thảo
Biên tập viên sách điện tử	: Nguyễn Tiến Sỹ
Trình bày sách	: Huỳnh Công Pháp Lê Việt Hoàng
Thiết kế bìa	: Đặng Nguyên Vũ

NHÀ XUẤT BẢN KHOA HỌC - CÔNG NGHỆ - TRUYỀN THÔNG
Website: https://nxbkhcntt.vn; book365.vn; ebook365.vn; ebook.gov.vn

Trụ sở chính: Tầng 6 - Tòa nhà Cục Tần số vô tuyến điện, 115 Trần Duy Hưng, P. Yên Hoà, TP. Hà Nội
Điện thoại: 024.35772139, 024.35772138

Chi nhánh TP. Hồ Chí Minh: Số 27 Nguyễn Bỉnh Khiêm, P. Sài Gòn, TP. Hồ Chí Minh
Điện thoại: 028.35127750, 028.35127751

Chi nhánh miền Trung - Tây Nguyên: 42 Trần Quốc Toản, P. Hải Châu, TP. Đà Nẵng
46 đường Y Jút, P. Buôn Ma Thuột, Đắk Lắk
Điện thoại: 0236.3897467/0262.3808088

ĐẠI SỐ TUYẾN TÍNH VÀ ỨNG DỤNG

TRONG TRÍ TUỆ NHÂN TẠO VÀ XỬ LÝ NGÔN NGỮ TỰ NHIÊN

(Sách chuyên khảo)

ISBN: 978-604-45-2616-4

GIÁ: 160.000Đ